42 55 54 59 49

Ára: 30 Ft

mikro számítógép magazin

A KISZOLGÁLTATOTT EMBER

1987/1

Előttünk a „Szivárvány”

Egyik munkatársunknak amerikai útja során alkalma nyílt az egyik szaklap szerkesztőségébe is ellátogatni. A baráti beszélgetés szakmai sikere, hogy a The RAINBOW (Szivárvány) c. amerikai szaklap és lapunk között cikkcsere indul.

A The RAINBOW® a Tandy (a HT—1080Z alapját képező TRS 80 Model I gyártója) által gyártott, TRS 80 Color elnevezésű, milliós példányban előállított gép lapja. Bár ez a gép — és a vele „rokon” Dragon — itthon méltatlanul alig ismert, a cikkcserének nemcsak az a jelentősége, hogy *az első, számítástechnikai lapok közötti amerikai—magyar kapcsolat.* Az amerikai lapban sok gépfüggetlen anyag is megjelenik. Ezek átvétele természetesen már közvetlenül hasznot is jelent számunkra.

Az amerikai lap rövid történetét a The RAINBOW® 1986. július, ünnepi számában, a tulajdonos-főszerkesztő Lonnie Falk cikkében olvashatjuk. A lapot öt éve alapították. Kezdetben egy másfél négyzetméter területű asztallap volt a „szerkesztőség”. Az alapító egyetlen alkalmazottal dolgozott, aki jelenleg a lap üzleti vezetője, a cikkcsere fő támogatója: Patricia H. Hirsch. Az akkori lapterjedelem 2 oldal volt.

Jelenleg a szerkesztőség saját, 2000 m² alapterületű, háromszintes épületében működik, 73 alkalmazottal. A lap 244 oldalas, sokszínű. A tulajdonos számos már folyóiratot is kiad: például a 80 ezres példányszámú VCR Magazine-t a videokazetta-használóknak és a PCM címűt az IBM PC használóknak. A The RAINBOW®-nek kazettamelléklete is van, melyben a lap fontosabb programjait értékesítik, mintegy havi 8000 példányban.

A következőkben rendszeresen közölni fogjuk az átvett cikkeket.

5. ÉVFOLYAM
1987/1. SZÁM

mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző

E számunkat szerkesztették:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Lindner László
(sakkprogramozás)

Petróczy Judit
(könyvek)

Simonyi Endre
(klub)

Varga András
(iskola-számítógép)

Címképünk:
Ramocsai Imri munkája

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levélcím:
1371 Budapest
Pf. 433.

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat

Felelős kiadó:
dr. Petrus György
igazgató

Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivataloknál
és a Posta Hírlapelőfizetési és Lapellátási Irodáján
(1900 Budapest V., József nádor tér 1.)
vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.

Megjelenik havonta
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, pf. 149.
és a Magyar Média
1932 Budapest, pf. 279.
86—253

Szikra Lapnyomda
Budapest (86-4622)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

Beszélgetések 2
Tájékoztató az 1987. évi Nemes Tihamér Tanulmányi Versenyről 6
Számítástechnika a fiataloknak 7
Mit tud a PROLOG? 9
Adok-veszek-cserélek 21
A PSION VU—3D rajzolóprogram 27
A kiszolgáltatott ember 35
Helyi hálózatok 36
Fantázia és tudomány? 38
Barátunk, a számítógép 40
A világ legnagyobb show-ja 48

ISKOLA — SZÁMÍTÓGÉP

Az ismeretlen C16 3
Spectrum az asztalban 4
Térkép 4
Középiskolai Számítástechnikai Konferencia 5

DIÁKROVAT

Credit 8

PROGRAMOZÁSTECHNIKA

Z80 programozási gyakorlatok 12
BASIC és gépi kód 14
Adalék a rendezéshez 15
Bináris keresőalgoritmus C64-re 16
Papírszélesítés 17
A VIC—20 tárcímei 19

μPROGRAMOK

A Beta basic és alkalmazása 22
Manó a Spectrumon 24
Új karakterek 25

SAKKPROGRAMOZÁS

Bitek és figurák 33
Gépi ellenfeleink 34

FÓRUM

Rendellenességek és rendetlenkedések 43

AZ OLVASÓ ÍRJA 44

PIAC 45

KÖNYVEK 46

HÍREK-ÉRDEKESSÉGEK 47

Beszélgetések

„Kora este a padon
Ülök künn a bástyán,
Tündöröm a csillagok
Néma fordulásán;
Kaszáscsillag, az öreg
A nyugati szélen
Éppen nyugszik; eleget
Ragyogott a télen."

(Tóth Árpád: Kaszáscsillag)

Ülök az MTESZ közgyűlésén és hallgatom az előadókat, fel-felkapom a fejem, ha egy ismerős, de még inkább ha olyan hozzászóló lép a dobogóra, akinek mondanivalója is van, és azt érdekesen, a figyelmet felkeltve tudja előadni.

Nincs szándékomban a közgyűlésről későbbi beszámolót írni, akit érdekelt, az a napilapokban, az MTESZ anyagokban elolvashatta. Egy örökzöld témáról írnék, mert amióta csak az MTESZ-ben dolgozom, azóta állandóan visszatérő gondja minden egyesületnek a hazai és nemzetközi konferenciák rendezése. Erről ezen a közgyűlésen is legalább két-három hozzászóló beszélt.

Arról van szó, hogy egy 4-5 napos tanácskozást ma már 2-4000 Ft körüli részvételi díj alatt gyakorlatilag nem lehet megrendezni (és ehhez jön a szállás meg az étkezés), hiszen a terembérleti díjak, a tolmácsberendezések, a tolmácsok, a sokszorosítás és a nyomda, de még a postai díjak emelkedése is alacsonyabb részvételi költség mellett a konferenciákat biztosan veszteségessé tenné. A helyzetet csak súlyosbította, hogy a központi apparátus költségeinek fedezésére az MTESZ minden konferenciát meg is „adóztat", rátéve a költségekre bizonyos százalékot, amelyet azután az egyesülettől elvon. Egyáltalán nem kell csodálkozni, hogy ezt a gyakorlatot az egyesületek kritizálják, hiszen ez az „adó" a jól dolgozó, sok konferenciát rendező egyesületeket — így az NJSZT-t is — sújtja, míg a kevésbé aktívakat, a főleg dotációból élőket szinte támogatja.

De nem erről szeretnék beszélni, inkább az érem másik oldaláról, amelyről a közgyűlésen nem hivatalosan esett szó. Arról ugyanis, hogy a sok-sok nehézség és gond, költségvetési bakugrások árán megrendezett konferenciák hasznát milyen mértékben élvezzük, és kiaknázzuk-e azokat a lehetőségeket, amelyek egy-egy nemzetközi konferenciában rejlenek.

Külföldi konferenciára járni jó, sokkal jobb, mint Pestre, Szolnokra, Szombathelyre vagy Szegedre utazni egy-egy nemzetközi találkozóra, mert világot lehet látni, érdekes emberekkel lehet beszélgetni, egy-egy különleges színház vagy mozi színesítheti az utazást, és persze venni is lehet néhány olyan dolgot, ami itthon nem, vagy csak nagyon drágán kapható. A SICOB, a Hannoveri Vásár, a müncheni Systems, a Lipcsei Vásár vagy pedig a bécsi IFABO lassan már magyar házi rendezvénynek is számít, és ez így van jól, ha többet szerepelünk a világban, ha többet látunk a világból, ha többet hasznosítunk az ott tapasztaltakból, gyorsabban haladunk, előbbre jutunk.

Legalább ilyennek, ha nem hasznosabbnak tartom az itthon rendezett nemzetközi konferenciákat is, amelyek — ha sikerül elegendő és színvonalas előadást és előadót szervezni — a világot hozzák el hozzánk: azoknak is megteremtik a szakma „híressége-ivel" a kapcsolat lehetőségét, akiknek nyelvi vagy sokszor anyagi problémák miatt a külföldi konferenciákon vagy kiállításokon való részvétel lehetősége nem adatik meg.

Mondom, erről is beszélgettünk az MTESZ-közgyűlés köreiben, azután egy hét múlva folytattuk a Teleteaching '86 nemzetközi konferencián. Azt nem értettük, ha idejön Budapestre — házhoz hozva a nemzetközi tapasztalatokat — 20, 50, 100 külföldi szakember, akiknek meg lehet hallgatni az előadását, akikkel a szünetben, az esti találkozókon, a kerekasztal-beszélgetéseken össze lehet ülni, akiket ki lehet kérdezni, akiket egy-egy intézetbe meg lehet hívni, hogy az ottani munkáról véleményt mondjanak, ezt miért nem használjuk ki eléggé. Ráadásul ez még pénzbe sem kerül, sőt ők fizetnek azért, hogy eljöhessenek.

Félreértés ne essék, a Teleteaching '86 konferencia jó volt, és ami a résztvevőket illeti, optimálisnak volt mondható; az előadásokon, bemutatókon ugyanis cca 80-100, sőt az első napon 150 hallgató vett részt. Így nagyon kellemes és alkotó légkörben tudtunk beszélgetni az oktatástechnológiáról a körülbelül húsz országból érkezett szakemberekkel. Jók voltak az előadások, és többségükben jók az előadók is, ezért majdnem mindegyik után vita alakult ki. Az előadásokat magyarra fordítottuk, hogy azok a hazai szakemberek is eljöjjenek és megértsék, akik esetleg nyelvi problémákkal küzdenek. Ennek ellenére a hazai hallgatók gyakorlatilag nem vettek részt a vitában, alig kérdeztek, az embernek órákon keresztül azt volt a benyomása — miután inkább a külföldiek vitatkoztak —, hogy nem egy budapesti, hanem egy párizsi vagy müncheni konferencián üldögél.

Amikor a konferencia rendezésébe kezdtem, azt hittem, hogy egy ilyen, nagyon sok újdonságot bemutató téma, mint a távoktatás, vagy — ahogyan a résztvevők a folytatásról szavaztak — távtanulás (distance learning) az oktatók, az oktatástechnikusok százait fogja csábítani, és a kb. 200 fős előadóterem kicsinek fog bizonyulni. Már a kezdet kezdetén úgy terveztük, hogy az anyagilag gyengén „elerésztett" oktatókat, illetve tanulókat kedvezményesen (1000 illetve 400 Ft), majd az utolsó előtti pillanatban meghirdetve, fizetés nélkül is beengedjük. Alig jöttek. Kiderült, hogy ebben néhány iskola vezetése is ludas, ahol pl. nem engedték el a tanárokat, mondván, hogy akkor ki tartja meg helyettük az órákat. Persze voltak olyan iskolák, ahonnan eljöhettek kikérésre vagy anélkül is. Én azt hiszem, hogy nem elegen, pedig az előadások mellett egy — előre nem szervezett — kiállítás is összejött: néhány alig ismert hazai és a legmodernebb külföldi (videolemez) oktatástechnikai eszközökből állt a bemutató.

Ha nem beszéltem volna a Teleteaching '86 előtt más egyesületek vezetőivel, akiknek a nemzetközi konferenciákkal kapcsolatban hasonló tapasztalataik voltak, akkor azt hittem volna, hogy itt az informatikával és különösen az oktatással kapcsolatos jelenségről van szó.

Azt kell mondanom, hogy a probléma súlyosabb: valószínűleg nem tudjuk megbecsülni és kihasználni azokat a lehetőségeket, amelyeket nagyon sok munka és fáradság árán az egyesületek végül is biztosítanak. Ezek a konferenciák és találkozók néhány ember ügyévé szegényednek, pedig az egyesületek, így a Neumann Társaság is mindent megtesz annak érdekében, hogy az érdekelteknek időben tudomására jusson a tapasztalatcsere — a tanulás és egyben a tanítás — lehetősége.

A Teleteaching '86 előtt, amikor a hazai előadókat szerveztem, nem egy barátom azzal utasította el a részvételt, hogy „mit keresnék én ott a sok külföldi sztár-előadó között azzal a szerény kis eredménnyel, amit be tudnék mutatni . . .". Szerencsére nem mindenki gondolkozott így, és a hazai „szerény" eredmények külföldi visszhangja azt bizonyította, hogy a mi előadásainkban is voltak olyan ötletek és eredmények, amelyekre bizony oda kellett figyelniük.

Kialakult az a véleményem is — egyoldalú információk alapján —, hogy a munkahelyek (és most itt az oktatási intézményekre gondolok elsősorban; ezért érdemes a jelenségre odafigyelni) a nem szervezett továbbképzést, tehát ahol nem „beiskolázásról" van szó, nem használják ki megfelelően. Én azt hittem, hogy az oktatásért felelős „főnökök" a részvételt az oktatástechnológiáért felelős szakembereknek kötelezővé teszik, majd a végén beszámoltatják őket, hogy az adott területen a hallottakból mit lehet hasznosítani.

Egy másik érdekes véleményt is hallottam, nevezetesen azt, hogy az embereknek általában nincs idejük konferenciára járni. A Teleteaching '86-on az előadások hét félnapon keresztül tartottak — gyakorlatilag reggel 9-től este 7-ig. „Aki a munkahelyén is helyt akar állni — mondják —, a gmk-ban vagy PJT-ben is keresni akar, annak a hét félnap megengedhetetlen kieső." A vélemény mögött sajnos több van, mint az az első pillanatban látszik, t.i. itt most arról van szó, hogy a napi munka mellett gyakorlatilag nincs, vagy egyre kevesebb idő jut az ismeretek felfrissítésére, új ismeretek megszerzésére. Azt hiszem, felesleges lenne részleteznem az ebből a helyzetből keletkező problémákat, hogy hova jut az a szakember, akinek nem jut ideje a szakmai továbbképzésre. Ez már nemcsak az egyén, hanem a társadalom problémája is. Az innováció ma divatos fogalom, amelyről nem beszélni kell, hanem tenni is kell érte valamit. Olyan körülményeket kell tehát teremteni, ami a szakembereket nemcsak arra kényszeríti, hogy dolgozzanak, hanem arra is, hogy fogékonyak legyenek az újra, tanuljanak. A munka eredményének elbírálásánál legyen követelmény annak újdonságtartalma, és hogy megfeleljen a mai minőségi igényeknek és technológiai előírásoknak. Ki kell alakítanunk a mai kor szakembermodelljét, aki rendszeresen tanul, képezi magát, minden újra fogékony, és a napi munkájában folyamatosan hasznosítani képes a tanultakat, természetesen a konferenciákon tanultakat is.

KOVÁCS GYŐZŐ

Az ismeretlen C16

KERNAL rutinok 4.

A KERNAL rutinok ismertetésének befejező részét egy helyreigazítással kezdem. A VECTOR rutin leírásában, amely a szeptemberi számban jelent meg, kétszer is tévesen 16 bájtos területre hivatkoztam 32 bájtos helyett. Ezen a területen ugyanis 16 címet kell elhelyezni, ezek mindegyike kétbájtos. A hibáért elnézést kérek.

Az itt következő hét rutint eddig nem használtam, működésüket csak a leírás miatt tanulmányoztam. Sajnos a SETTMO esetében a rendelkezésemre álló irodalom ellentmondásai miatt csak sejtem a valóságot. Remélem, hogy akinek szüksége van ennek a rutinnak a használatára, hozzá tud jutni a szükséges pontos információhoz.

CINT $FF81 65409

Alaphelyzetbe állítja a képernyős szerkesztőt és a videoregisztereket. (A VC20-on ez a KERNAL rutin hiányzik.)

IOINIT $FF84 65412

Alaphelyzetbe állítja a beviteli/kiviteli rendszert. (A VC20-on ez a KERNAL rutin hiányzik.)

SETMSG $FF90 65424

Ezt a rutint legtöbbnyire az operációs rendszer rutinjai használják egy-egy képernyőre írandó üzenet kiadása előtt. Egy jelzőt állít be aszerint, hogy hibaüzenetről vagy egyéb információról van szó. A kívánt értéket a hívás előtt az A regiszterbe kell tölteni, hibaüzenet esetén $80-at, egyébként $40-et. Az a rendszerváltozó, amelybe a rutin az A regiszter tartalmát leteszi, egyúttal a futási mód jelzője is. Ennek negatív értéke esetén, vagyis hibaüzenet kiadása után, a BASIC interpreter közvetlen módba megy át.

MEMTOP $FF99 65433

Az átviteljelző (C) bit értékétől függően beállítja vagy kiolvassa a BASIC által használható RAM terület felső határára mutató változó tartalmát. C=1 esetén a mutató tartalma az X/Y regiszterpárba kerül, ha C=0, a regiszterek tartalma töltődik a mutatóba. Az X regiszterhez tartozik a változó alacsonyabb helyiértékű bájtja, Y-hoz a magasabb helyiértékű.

MEMBOT $FF9C 65436

Szerepe megegyezik a MEMTOP-éval, de a BASIC RAM alsó határára mutató változó értékét állítja, illetve olvassa. A BASIC interpreter inicializálásakor a MEMTOP és MEMBOT által kezelt változókból kerülnek az információk a megfelelő nullalapos mutatókba.

SETTMO $FFA2 65442

Ezzel a rutinnal kapcsolatosan a forrásmunkáimban teljes a káosz. (Zárójelben jegyzem meg, hogy ezek többsége a C64-re vonatkozik, de KERNAL rutinokról lévén szó, az erre vonatkozó adatoknak C16-ra is érvényeseknek kell lenniük.)

A rutin nem csinál mást, mint az A regiszter tartalmát elhelyezi egy rendszerváltozóba, melynek neve: time-out flag, szabad fordításban időtúllépés-jelző. Hívása előtt a kívánt értéket az A regiszterbe kell tölteni. A kapcsoló állapotát a 7. (bal szélső) bit jelzi. Azt, hogy hogyan, csak sejtem, ugyanis az erre vonatkozó adatok is ellentmondanak egymásnak.

Dr. Úry László Commodore 64 című könyvének II. kötetében a következő olvasható: „Ha a rutin meghívásához az A regiszter 7. bitje magas, a time-out jelző beállítódik. Ha a bit alacsony, a jelző kikapcsolódik." Ezzel szemben Erdős Iván Commodore 64 Assembly című könyvében a következőt találtam: " . . . a 7. bit magas — törlés . . . alacsony — beállítás". (Mindkét könyv az LSI ATSZ kiadványa, az utóbbinak a lektora dr. Úry László.) Ezeken kívül még egy — VC20-ról szóló — könyvben találkoztam a jelző beállításának miértjével, az szerint az alacsony érték jelenti a bekapcsolt állapotot. A leírások többsége a time-out kapcsolót a soros busszal hozza kapcsolatba, három helyen — a fentebb idézett két könyvben és egy német nyelvű folyóiratban — az IEEE buszra való utalást találtam. Tapasztalataim alapján én is az utóbbira tippelek.

IOBASE $FFF3 65523

A beviteli/kiviteli modulok kezdőcímét tölti az X/Y regiszterpárba, alacsony/magas helyértékű bájtsorrendben. Ez egy géptípustól függő állandó a C16-on $FD00. (VC20-on: $9110, C64-en:$DC00.)

BARNA LÁSZLÓ

C16 orgona

Ez az egyszerű kis zenei program orgona jellegű, folyamatos és változó időtartamú hanghatásokat vezérel billentyűzet segítségével.

Az adott billentyű lenyomásakor a megfelelő hang megszólal, és addig szól, amíg egy másik hangbillentyűt le nem nyomunk. Zenei szünetet a szóközbillentyűvel lehet létrehozni, de a hangot elhallgattatja a még fel nem használt billentyűk lenyomása is. A program tartalmaz egy védelmet: hibaüzenet vagy meggondolatlan BREAK esetén az újraindítás után a STOP? kérdésre adott I válasz inicializálja a hanggenerátort. Egyszerre két hangbillentyűt lenyomva érdekes hanghatások jönnek létre, bár a „hangszer" monofonikus.

A négyoktávnyi egész hangok kibővíthetők a félhangokkal is: erre a billentyűzet lehetőséget ad (4×12 db), csak az alsó sorban a SHIFT-et kell átlépni. Így azonban a kezelés lényegesen bonyolultabbá válik.

GULYÁS ANDRÁS

```
  5 PRINT"■■■■■■■■■■■■■■■■■■ ORGONA ■"
 10 PRINT "■■■■AZ 1 - 8"
 20 PRINT "■■■■■Q - I"
 30 PRINT "■■■■■A - K"
 40 PRINT "■■■■■Z - ,"
 50 PRINT "■■■■■■■■■■BILLENTYUKKEL LEHET ZENELNI"
 51 PRINT"■■■ SZOKOZ SZUNETET AD"
 52 PRINT"■■■BEFEJEZES:RUN/STOP"
 53 PRINT"■■■A EZUTAN SZOL A HANG, UJRA"
 54 PRINT"■■INDITASSAL A STOP?-RA ADOTT"
 55 PRINT"■■'I' ELHALLGATTATJA"
 56 PRINT"■■JO SZORAKOZAST!"
 60 VOL 6
 70 DATA 596,0,0,0,0
 80 DATA 917,929,939,944,953,961,967.5,971.5,0,0,0,0,0,0,0,0
 90 DATA 596,453,345,685,854,704,739,770,917,798,810,0,571,516,0,0,810,864
100 DATA 643,881,911,383,834,262,897,169
110 DIM N(47): FOR I=1 TO 47: READ N(I): NEXT I
120 PRINT "■■■■STOP?":GETKEY W$:IF W$="I" THEN GOSUB 220
130 PRINT "■■OK,LEHET ZENELNI!"
140 GETKEY A$: X=ASC(A$)
150 IF X<44 OR X>91 THEN GOTO 140
160 B=ASC(A$): K=B-43
170 IF N(K)<>0 THEN SOUND 1,N(K),60000
180 DO: GETKEY A$: X=ASC(A$)
190 IF X<44 OR X>91 THEN SOUND 1,N(K),0:GOTO 140
200 LOOP WHILE ASC(A$)=B
210 SOUND 1,N(K),0: B=ASC(A$): K=B-43: GOTO 170
220 FOR L=1 TO 47:SOUND 1,N(L),0:NEXT L
230 RETURN
```

Spectrum az asztalban

A Pécs város másik végén tanító fiatal gimnáziumi tanárt, Sebestyén Zoltánt a lakása szomszédságában lévő általános iskola hívta tanítani. Évekig gondolkozott, aztán megunva a közlekedésre pocsékolt sok időt, komolyan foglalkozni kezdett az ajánlattal. Matematikatanár lévén úgy döntött, hogy ha vesz az iskola egy számítógépet, átjön. Az igazgató azonnal megvett egy ZX-Spectrumot, pedig akkor még 40 ezer forintba került. Így aztán Sebestyén Zoltán is általános iskolai tanár lett a szomszédban lévő iskolában. De ettől kezdve csak este 10 óra után látta a családja: minden este a számítógépet vigyázta a gyerekek kezében. Elzárni nem volt szíve, sőt inkább a sajátját is bevitte mindig, s közben egyfolytában azon töprengett: hogyan tudna kiszabadulni ebből a saját maga ásta csapdából, s visszanyerni az esti szabadságát.

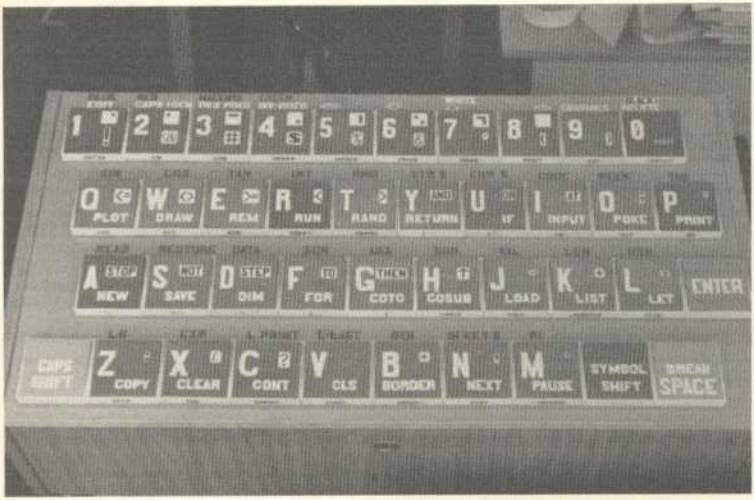
Ekkor villant fel benne az ötlet, hogy a ZX-Spectrum billentyűzetét kivezeti egy asztallapra. Az ötletet tett követte, s egy gyermekíróasztal felületére szerelt ajtócsengő-kapcsolókkal elkészítette az új Spectrum-billentyűzetet. A gépet magát a biztonsági zárral ellátott fiókba rakta, a billentyűzet és a gép közötti kapcsolatot pedig a gép buszcsatlakozójára ráhúzható illesztővel biztosította. Ebben mérnök barátja nyújtott segítséget.

Azóta a Spectrum az iskola folyosóján áll. Reggel fél nyolckor bekapcsolják, s este, az iskolából utoljára távozó tanár kapcsolja csak ki. Délelőtt, a tanítási órák alatt a gép a szülőké; a gyerekeknek kihirdették, hogy küldjék be a számítástechnika iránt érdeklődő szüleiket. A szünetekben persze, ha üres a gép, a gyerekek versenyt rohannak, hogy ki gépeljen be egy pár perc alatt is beírható programot. Délután már a különböző szakkörök használják a gépet, előre ütemezett módon. Este ismét kötetlenül a gyerekeké: előre feliratkozhatnak, hogy ki mikorra igényli.

A nagy billentyűzet újszerű géphasználatot vezetett be. Játékoknál hárman-négyen is hozzáférnek a géphez, s „munkamegosztás"-ban nyomogatják a billentyűket, miközben igazi csapatszellem: egy mindenkiért — mindenki egyért alakul ki köztük. Ennek a megoldásnak a másik új alkalmazása a gép billentyűzetének falra akasztható-sága: a falon táblaként elhelyezkedő billentyűzet az egész osztály számára jól látható, és remekül segíti az oktatást.

A mintegy tízezer forintba kerülő billentyűzet iránt érdeklődők megkereshetik az alkotót, aki megfelelő igény esetén vállalkozik további billentyűzetek készítésére. Címe: Sebestyén Zoltán, 7632 Pécs, Testvérvárosok tere, általános iskola. Telefon: (72) 10-993.

Az asztalba épített ZX-Spectrum billentyűzet



PRIMO

Térkép

A program segítségével játékosan gyakorolhatjuk a földrajzot. A következő feladatokat menüből választhatjuk ki.

1. A megjelölt pont felismerése. A térképen egy földrajzi pontot takar el egy négyzet. Ennek a pontnak a nevét kell válaszként beírni. A gép számolja és kiírja a feladványokat, valamint a sikeres megfejtéseket.

2. Utazás hibajelzéssel. A gép a rendelkezésére álló adathalmazból kiválaszt egy úticélt, amelyhez az É, a D, a K és az N betűk segítségével el kell vezetni a képernyő bal alsó részén lévő pontot. A pont sebessége a SHIFT gombbal állítható. A képernyő jobb felső részén felirat jelzi, hogy közeledünk-e a célhoz vagy távolodunk tőle. A gép számolja és kiírja a távolodások számát.

```
1 REM TERKEP                    (
9 U$="                          "
10 RESTORE : CLS : PRINT : PRINT
     "MI A FELADAT?":PRINT
12 PRINT"1 A MEGJELOLT PONT FELISMERESE"
13 PRINT"2 UTAZAS HIBAJELZESSEL"
14 PRINT"3 UTAZAS HIBAJELZES NELKUL"
15 PRINT"4 ELLENORZES"
19 PRINT"5 BEFEJEZTUK" : PRINT
20 INPUT L : CLS : IF L>5 OR L<1 THEN10
30 ON L GOTO 7300,7350,7350,7000,40
40 CLS : END
100 REM RAJZOLO .....................
109 FOR K=1 TO 7
110 READY1,X1 : Y1=Y1/2 : X1=X1/2
112 READY2,X2 : Y2=Y2/2 : X2=X2/2
118 IF X2=0 THEN 130
119 GOSUB 990
122 X1=X2 : Y1=Y2
128 GOTO 112
130 NEXT K
132 RETURN
149 REM IDE KELL IRNI A HATAR, A DUNA,
290 REM A TISZA ES A BALATON KIRAJZOLA-
310 REM SAHOZ SZUKSEGES ADATOKAT
990 REM KET PONTON ATMENO EGYENES .....
1003 IF X2=X1 THEN LR=1 : GOTO 1020
1004 Z=(Y2-Y1)/(X2-X1)
1005 IF Y2=Y1 OR ABS(Z)<1 THEN LR=1
       ELSE LR=1/ABS(Z)
1007 IF X1>X2 THEN 1010
1008 LP=1 : GOTO 1012
1010 LP=-1
1012 Q=X1 : W=X2
1013 FOR X=Q TO W STEP LP*LR
1014 Y=Z*(X-X1)+Y1
1017 SET(X,Y)
1018 NEXT X
1019 GOTO 1032
1020 IF Y1>Y2 THEN 1024
1022 LP=1 : GOTO 1025
1024 LP=-1
1025 Q=Y1 : W=Y2
1026 FOR Y=Q TO W STEP LP*LR
1029 SET(X1,Y)
1030 NEXT Y
1032 RETURN
5000 REM A PONT VEZETESE.............
5002 QQ$=INKEY$ : IFQQ$<>"" QQ=ASC(QQ$)
5003 IF QQ$="" THEN 5002
5004 IF QQ$="É" THEN Y=Y+10
5005 IF QQ=100 THEN Y=Y-1
5006 IF QQ$="D" THEN Y=Y-10
5007 IF QQ=107 THEN X=X+1
5008 IF QQ$="K" THEN X=X+10
5009 IF QQ=110 THEN X=X-1
5010 IF QQ$="N" THEN X=X-10
5011 IF QQ=96  THEN Y=Y+1
5012 IF X>210 THEN X=210
5013 IF X<1   THEN X=1
```

```
5014 IF Y>132 THEN Y=132
5015 IF Y<1   THEN Y=1
5016 IF (X=X9 AND Y=Y9) THEN 5002
5018 IF POINT(X9,Y9)=0 AND X9>0 THEN
       SET(X9,Y9) ELSE RESET(X9,Y9)
5020 IF POINT(X,Y)=0 THEN
       SET(X,Y) ELSE RESET(X,Y)
5022 X9=X : Y9=Y : QQ=0
5113 RETURN
6000 REM ADATKEZELO .....................
6005 K=RND(9) : V=RND(10)
6007 ON K GOTO 6011,6012,6013,6014,6015,
       6016,6017,6018,6019
6011 RESTORE 6100 : GOTO 6030
6012 RESTORE 6200 : GOTO 6030
6013 RESTORE 6300 : GOTO 6030
6014 RESTORE 6400 : GOTO 6030
6015 RESTORE 6500 : GOTO 6030
6016 RESTORE 6600 : GOTO 6030
6017 RESTORE 6700 : GOTO 6030
6018 RESTORE 6800 : GOTO 6030
6019 RESTORE 6900 : GOTO 6030
6030 FOR K=1 TO V
6032 READ QQ$,Y,X : Y=Y/2 : X=X/2
6040 NEXT K
6045 RETURN
6100 DATA GYOR,183,97
6200 DATA KOMLO,50,136
6300 DATA SZEKESFEHERVAR,138,144
6400 DATA ESZTERGOM,191,164
6500 DATA PANNONHALMA,169,116
6600 DATA KOSZEG,158,32
6700 DATA OZD,230,259
6800 DATA EGER,201,266
6900 DATA BERETTYOUJFALU,140,337
6910 DATA 0,0,0
7000 REM ELLENORZES .....................
7001 GOSUB 7550
7002 GOSUB 100
7003 RESTORE 6100
7004 READ N$,Y,X
7005 X=X/2 : Y=Y/2
7006 IF N$="0" THEN GOTO 10
7007 PRINT#0,0,U$;U$;U$
7008 PRINT#0,0,N$
7009 GOSUB 7100
7030 QQ$=INKEY$ : IF QQ$="" THEN 7030
7038 GOSUB 7100
7044 GOTO 7004
7100 REM JELOLO .........................
7109 FOR I=X-3 TO X+3
7110 FOR K=Y-3 TO Y+3
7112 IF POINT(I,K)=0 THEN SET(I,K) ELSE
       RESET(I,K)
7120 NEXT K : NEXT I
7122 RETURN
7300 REM A MEGJELOLT PONT FELISMERESE .
7301 HE=0 : HI=0
7302 GOSUB 7500
7303 GOSUB 100
7304 GOSUB 6000
7306 GOSUB 7100
7308 PRINT#0,0,U$;U$;U$
7309 PRINT#0,0,"MI A NEGYZETTEL ";
       "LETAKART HELY NEVE?"
7311 INPUT Q$ : PRINT#0,0,U$;U$;U$;U$
7312 IF Q$=QQ$ THEN HE=HE+1:PRINT#1,0,
       "H E L Y E S !" : GOTO 7314
7314 PRINT#1,0,"H I B A S ! HELYESEN: "
       ;QQ$ : HI=HI+1
7316 PRINT HE+HI;" JATEKBAN";HE;
       " TALALATOD VOLT."
7318 Q$="N" : INPUT "FOLYTATJUK";Q$
7319 GOSUB 7100
7320 FOR I=0 TO 3 : PRINT#I,0,U$;U$;U$
7321 NEXT I
7324 IF (Q$="I" OR ASC(Q$)=105) THEN
       7304 ELSE GOTO 10
7350 REM UTAZAS RAVEZETESSEL .........
7351 GOSUB 7520
7352 GOSUB 100
7354 GOSUB 6000
7355 X1=X:Y1=Y:D1=0:X=5:Y=5:HI=0
7357 SET(5,5)
7358 PRINT#0,0,U$;U$;U$
7359 PRINT#0,0,"AZ UTICEL: ";QQ$
7360 GOSUB 5002
7361 RESET(5,5)
7362 D=SQR((X1-X)^2+(Y1-Y)^2)
7363 IF D1=0 THEN 7370
7364 IF D<D1 THEN PRINT#0,30,
       "KOZELEDIK"
7366 IF D>D1 THEN PRINT#0,30,
       "TAVOLODIK" : HI=HI+1
7370 D1=D : IF D>1 THEN 7360
7377 PRINT#0,0,"MEGERKEZTUNK!";U$;U$;U$
7379 GOSUB 7100
7380 IF L<>3 THEN PRINT#1,0,HI;
       " ALKALOMMAL TAVOLODTAL ";
       "AZ UTICELTOL."
7388 Q$="N" : INPUT"FOLYTATJUK";Q$
7389 GOSUB 7100
7390 FOR I=0 TO 2:PRINT#I,0,U$;U$;U$;U$
7392 NEXT I
7393 IF (Q$="I" OR ASC(Q$)=105) THEN
       7354 ELSE GOTO 10
7500 REM TAJEKOZTATO ...................
7502 CLS
7504 PRINT :PRINT
7506 PRINT"A TERKEPEN EGY 'FOLDRAJZI ";
       "PONTOT' TAKARUNKEL EGY NEGYZ";
       "ETTEL."
7509 PRINT"A NEGYZET KOZEPEN LEVO 'PO";
       "NT' NEVET KELL"
7511 PRINT"BEIRNI."
7512 PRINT"(AZ ELSO BETU NAGY, A TOBB";
       "I KICSI.)
7514 GOTO 7600
7520 PRINT : PRINT
7521 PRINT"A KEPERNYO BAL ALSO RESZE";
       "SZEN LEVO PONTOT"
7523 PRINT"KELL AZ UTICELHOZ VEZETNI,";
       " AZ 'E', A 'D', A 'K' ES AZ ";
       " 'N' BETUK SEGITSEGEVEL. A BE-"
7528 PRINT"TUK AZ EGTAJAKAT JELOLIK."
7529 PRINT"A PONT SEBESSEGE A 'SHIFT'";
       "-TEL ALLITHATO."
7531 PRINT"A KEPERNYO JOBB FELSO RESZ";
       "EN LEVO"
7533 PRINT"FELIRAT JELZI, HOGY KOZELE";
       "DUNK A CELHOZ, VAGY TAVOLOD";
       "UNK TOLE."
7536 PRINT : GOTO 7600
7550 PRINT : PRINT
7556 PRINT"A TERKEPEN EGY 'FOLDRAJZI ";
       "PONTOT' TAKARUNKEL EGY NEGYZ";
       "ETTEL."
7559 PRINT"A NEGYZET KOZEPEN LEVO 'PO";
       "NT' NEVET"
7561 PRINT"KIIRJUK."
7564 PRINT"AZ ELLENORZEST EGY KARAKTE";
       "R BEIRASAVAL"
7566 PRINT"FOLYTATHATJUK."
7600 PRINT : PRINT"A PROGRAM EGY KARA";
       "KTER BEIRASAVAL FOLYTAT-HATO."
7604 QQ$=INKEY$ : IF QQ$="" THEN 7604
7605 CLS : PRINT CHR$(6)
7606 RETURN
```

3. Utazás hibajelzés nélkül. A program működése megegyezik az előzővel, de a gép nem írja ki a távolodások számát.

A programot Magyarország térképéhez készítettem, de más térképhez is használható. A működéséhez szükséges adatokat az érdeklődőknek levélben megküldöm. Ezeket DATA utasításokban a következő módon kell elhelyezni.

A 133-as sortól a 989-es sorig kell beírni a határ, a Duna, a Tisza és a Balaton vonala töréspontjainak koordinátáit, a 6100-tól a 6900-ig tartó sorokban lévő értékekkel összehangolva. Az értékek között 0 nem lehet, mert a „0,0" koordinátaérték az adott vonal rajzolásának befejezését jelzi a gépnek.

A 6000-es sortól egy gyors adatkezelő rész található, amely 9 csoportban tíz adategységet kezel. A földrajzi pontok adatait úgy kell kiegészíteni, hogy a következő sorok között tíz-tíz adategység (megnevezés és koordináták) legyen: 6100—6109, 6200—6209, 6300—6309, 6400—6409, 6500—6509, 6600—6609, 6700—6709, 6800—6809, 6900—6909. A 6910-es sorban levő értéket meg kell tartani.

A programot más térképre átírva, a méretarányokat a 110-es, 112-es, 6032-es és 7005-ös sorokban kell módosítani.

SOMOGYI GYÖRGY

Középiskolai Számítástechnikai Konferencia

Fonyód, 1986. augusztus

Már hagyományosnak számít, hiszen ezúttal negyedszer rendezték meg ezt a szakmai találkozót. A négynapos konferencia szervezői és lebonyolítói — a fonyódi Karikás Frigyes Gimnázium, Szakközépiskola és Kollégium, a TIT Budapesti Szervezete matematikai szakosztálya és az NJSZT Somogy Megyei Szervezete — munkáját dicséri, hogy az idén több mint százötvenen vettek részt közel 120 iskolából, s rajtuk kívül az előadók és a kiállítók.

A megnyitót dr. Komáromi József, a Fonyód Nagyközségi Tanács elnöke tartotta, aki nemcsak ízig-vérig lokálpatrióta, de a számítástechnikai oktatás lelkes támogatója is. A további előadók — tanárok, szakfelügyelők, ipari szakemberek — módszertani, oktatástechnikai, hardver- és szoftverkérdésekről, gépekről és rendszerekről tartottak előadásokat. Izelítőként felsorolunk néhány érdekesebbet közülük.

Csőke Lajos (Eger): A számítástechnikai oktatás tapasztalatai

Kertész Ádám (Budapest): Programozási módszerek

Dr. Paronyai Gábor (Baranya m.): Középiskolai számítástechnikai szakkörök

Hajdú János (Csongrád m.): Z80 Assembler

Horváth István (Fonyód): Lokális hálózat HT gépre

Nyirati László (Székesfehérvár): FORTH programozás

Dr. Balla László (Zala m.): A középiskolai számítástechnika módszertana

Jelentősége miatt kiemelkedett a programból a számítástechnikai oktatás kérdéseiről folytatott vita, amely a Tudományszervezési és Informatikai Intézet igazgatójának beszámolóját követte. Páris György ugyanis a számítástechnikai oktatási program jelenlegi eredményeiről és jövőbeni folytatásáról beszélt, majd a számítástechnikát oktató pedagógusok ismertették sikereiket és elmondták gondjaikat.

A konferenciával egyidejű bemutatón a résztvevők „testközelben" vizsgálhatták meg a hazai iskolaszámítógép-gyártók termékeit, a HT—1080Z/64, a HT—3080C, a Primo, a TV-Computer, az Aircomp és a Homelab számítógépeket.

Legfontosabb talán, hogy a fonyódi találkozó a középiskolai számítástechnika olyan fórumává vált, ahol a pedagógusok szelektív kapcsolatot alakíthatnak ki számítástechnikusokkal és más intézményekben dolgozó kollégáikkal, így lehetőségük nyílik a közvetlen tapasztalatcserére. Ezért a rendezők és a résztvevők egyaránt a hagyomány folytatását szorgalmazzák.

ERDÉSZ ISTVÁN

TÁJÉKOZTATÓ

az 1987. évi Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Versenyről

A Művelődési Minisztérium — a Neumann János Számítógéptudományi Társaság és a Magyar Kommunista Ifjúsági Szövetség Központi Bizottsága szervezésében — az 1986/87-es tanévben is megrendezi a Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Versenyt.

A verseny tárgya, követelményei:

— számítástechnikai és programozási alapismeretek,

— számítástechnika-alkalmazási alapismeretek a középiskolai közismereti tárgyak, az egyszerű adatfeldolgozás stb. témaköréből,

— a feladatok rendszerszemléletű feldolgozása, algoritmusok célszerű kialakítása, a hibás adatok bevitele elleni védelem, egyszerű és kényelmes kezelői felület kialakítása.

A versenyen a középfokú iskolák III. és IV. osztályos tanulói vehetnek részt. A versenyre legkésőbb

1987. január 9-ig

nevezhetik be az iskolák tanulóikat a következő címen:

Neumann János Számítógéptudományi Társaság
NEMES TIHAMÉR verseny
1368 Budapest 5, Pf. 240

A nevezésnek tartalmaznia kell

— az iskola nevét, címét és telefonszámát,

— a részt vevő tanulók számát évfolyamonként,

— a versenyért felelős tanár nevét.

A nevezést ajánlottan kell postázni.

Az első — az iskolában megtartandó — versenyforduló ideje

1987. február 10. kedd, 15.00 óra

Az első forduló dolgozatainak megíratásával és javításával kapcsolatos költségeket az iskolának kell viselnie.

A második, döntő fordulót Budapesten tartjuk meg

1987. április 11-én, szombaton.

Az időpontot és helyet később közöljük. A döntő résztvevőinek utazási költségét a küldő iskolának kell fedeznie.

A Nemes Tihamér OKSZTV első tíz helyezettje mentesül a felvételi vizsga alól és megkapja a maximális 30 pontot a matematika tárgyból, ha olyan felsőfokú oktatási intézménybe jelentkezett felvételre, amelyben a matematika felvételi tárgy. Ebben az esetben viszont matematikából érettségi vizsgát kell tenniük.

További kiemelkedő teljesítményért az NJSZT és a KISZ KB különjutalmakat ad.

A felkészülés megkönnyítésére a versenybizottság a következő könyveket ajánlja a versenyzők figyelmébe:

1. Alapvető számítástechnikai és programozási ismeretek

1.1 Szlávi—Zsakó: Módszeres programozás. Bp., 1986. MK.

1.2 Lőcs: A BASIC és a Kíváncsi + Feladatgyűjtemény (Középiskolai szakköri füzetek). Bp., 1985., 1986. TK.

1.3 Wirth: Algoritmusok + adatstruktúrák = programok. Bp., 1982. MK.

1.4 Csépai: A számítástechnika alapjai. Bp., 1985. MK.

2. Nyelvek

2.1 Gordon—Körtvélyesi—Sós—Székely: Pascal programozási nyelv. Bp., 1982. SZÁMALK.

2.2 Bakos: Pascal PC-seknek. Bp., 1986. MK.

2.3 A LOGO programozási nyelv. Bp., 1986. MK.

2.4 Lásd 1.2

3. Alkalmazások

3.1 Newman—Sproull: Interaktív számítógépes grafika (első fele). Bp., 1985. MK.

3.2 Kaufmann: Az optimális programozás (Modellek és módszerek). Bp., 1968. MK.

A Nemes Tihamér OKSZTV versenybizottsága

PÁLYÁZAT

A KISZ KB Középiskolai és Szakmunkástanuló Tanácsa és a Skála-Coop Számítógépes programpályázatot hirdet középiskolás diákok és pedagógusok számára.

A pályázat célja, hogy az ATARI típusú iskola-számítógép használatát megkönnyítse a tanórán és tanórán kívül, ezért a Skála a pályázaton elfogadott programokat MEGVÁSÁROLJA, és a gépek mellé egységcsomagban adja.

A pályázati szoftver vásárlási díja annak értékétől függ; néhány ezertől több tízezer forintig terjedhet.

A pályázatra más géptípusra — Commodore 64, 16 — írt programot is beküldhettek, ha az az ATARI számítógépre átírható. Átírásról a pályázat kiírói gondoskodnak.

A programokat az alábbi témakörökben várjuk:

— oktatást segítő programok (demonstrációs, oktató, feleltető),

— mozgalmi munkát, szabadidős tevékenységet segítő programok (pl.: képújságprogram, játékprogramok stb.),

— iskolai adminisztrációt segítő programok.

A programokat 1987. március 15-ig folyamatosan lehet beküldeni a következő címre:

KISZ KB KSZT
„ATARI PÁLYÁZAT"
1388 Budapest Pf. 72.

Beküldendő a program mágneses adathordozón (lemez vagy kazetta) és a program használatához szükséges leírás. A pályamunkákat szakemberekből álló zsűri bírálja el.

A megvásárlás mellett a SKÁLA által biztosított díjak:

1. díj ATARI 800 XL számítógép teljes kiépítéssel 1 éves használatra,

2. díj ATARI 800 XL számítógép teljes kiépítéssel féléves használatra,

3—6. tárgyjutalom.

Ezen felül a díjazottak meghívást kapnak a KISZ KB KSZT nyári számítástechnikai táborába, a közösségek részt vesznek a KSZT μ-klub pályázatán is.

KISZ KB
Középiskolai
és Szakmunkástanuló
Tanács
SKÁLA-COOP

Számítástechnika a fiataloknak

Hol tart ma a számítástechnika terjesztése az ifjúság körében Mongóliában? E kérdés nyomába indultunk Ulánbátorban, és így jutottunk el a Mongol Forradalmi Ifjúsági Szövetség Központi Bizottságának székházába.

— A mongol fiatalok számára az 1986-os év jelzi a számítógépes korszak kezdetét. A dolog tulajdonképpen Todor Zsivkov elvtárs, a Bolgár Kommunista Párt főtitkára 1984. évi látogatásával kezdődött — mondja Purevacem elvtárs, a dolgozó fiatalok osztályának munkatársa. — Zsivkov elvtárs felfigyelt arra, hogy itt nagy szükség van mikroszámítógépekre, és intézkedett bolgár gyártmányú berendezések szállításáról. 1986 májusában megérkezett 10 darab Pravec—82 típusú, az Apple II-vel kompatibilis gép a bolgár Dimitrovi Kommunista Ifjúsági Szövetség ajándékaként. Mindjárt szerveztünk egy háromnapos bemutatót Ulánbátor ifjúsági műszaki házában, és ott a fiatalok ezrei próbálhatták ki a gépeket. Hatalmas sikerük volt.

— *Jelenleg hol vannak ezek a gépek, és hogyan juthatnak hozzájuk a fiatalok?*

— Itt, a Központi Bizottság székházában rendeztünk be a gépek számára egy szobát. Lehetővé tesszük, hogy bárki ingyen használhassa őket, ezért a helyiséget állandóan nyitva tartjuk. Megszerveztük az éjjel-nappali konzultációt is biztosító teremfelügyeletet.

— *Ezek a mikroszámítógépek az elsők között érkeztek Mongóliába. Hogyan készítették fel a konzultációt biztosító fiatalokat?*

— Bár e gépek beérkezésekor valóban még csak pár darab, néhány hónappal „idősebb" mikroszámítógép volt az országban, már készültünk ezek fogadására. Jómagam például a szófiai egyetemen végeztem, és ott természetesen a mikroszámítógépek alkalmazását is tanultam. Ezenkívül a dimitrovi komszomol a Pravec—82 gépekkel egy szakembert is küldött egy évre. Az ő feladata a gépközeli információk átadása, az operatív konzultáció, valamint a számítástechnikai kultúrának az ifjúság körében való terjesztése.

— *Van-e valamilyen szervezett formája az ismeretátadásnak?*

— Folyamatosan indítunk tanfolyamokat, és szakkönyvek megjelentetésére is gondolunk. Kiadásunkban már meg is jelent az első füzet, amely a Pravec—82 kezelésének alapjait tartalmazza, természetesen mongol nyelven.

A Mongol Forradalmi Ifjúsági Szövetség székháza. Itt van a mikroklub

Pravec—82 típusú gépek — oktatókabinetben

— *Milyen terveik vannak a számítástechnikának az ifjúság körében való terjesztésére?*

— Ez két vonalon történik. Az állami irányvonal az iskolákon keresztül valósul meg, a művelődési minisztérium felügyelete alatt. Ennek keretében még az idén 30 szovjet mikroszámítógépet kapunk — valószínűleg az Apple II-vel kompatibilis Agat típusból. Az Állami Pedagógiai Intézet pedig már készíti elő a számítástechnika tantárgy bevezetését. Ez valószínűleg abba az iskolareformba épül bele, melynek keretében a jelenleg tízosztályos iskolai képzés 11 osztályosra alakul át. A másik vonal a számítástechnikai kultúra terjesztésének társadalmi irányvonala, melyet ifjúsági szövetségünk szervez. Itt jól kiépített bázisra támaszkodhatunk, mivel minden nagyobb városban vannak már ifjúsági műszaki házak. Ezekben szerelünk fel számítógéptermeket, melyeket a szocialista országokból beszerzendő mikroszámítógépekkel akarunk felszerelni. Az ajándékba kapott 10 mikroszámítógép az első lépést jelentette. Ezzel ráléptünk egy hosszú útra, melyen a további métereket — vagy kilométereket — most készítjük elő.

DR. BROCZKÓ PÉTER

Credit

Karakterkészlet editor

DIÁK C–64 ROVAT

A program a BASIC munkaterület alsó határa felett 2 k-val (4096 dec) kezdődik. Felhasználói karakterkészletek tervezésére és mágneslemezen való tárolására készült. Főbb funkciói tehát: adott kódú karakter képének megváltoztatása, lépkedés egyik karakterről a másikra, ugrás bárhova a karakterkészleten belül, karakterkészlet kimentése lemezre, karakterkészlet betöltése lemezről.

Apró és könnyen hibának vehető trükk a program 1000-es sora, melyre csak egyszer történik hivatkozás egy IF—THEN utasítással, a 210-es sorban. Ettől, illetve ennek a sornak a helyére, 1000-től kerülhetnek esetleges bővítő programrészek, mivel e sorra csak akkor ugrik a program, ha olyan gombot nyomunk le, amit nem tud értelmezni.

A program a kurzorvezérlő gombokat használja a kurzornak adott karakteren belüli vezérlésére: a * gombot a pont kigyújtására, a DEL-t a kikapcsolására, a CLR-t a karakter törlésére, a HOME gombot a kurzor visszapozicionálására, a "+"-t és a "–"-t előre-hátra lépésre a karakterkészletben, az S-t és az L-t kimentésre, illetve betöltésre, a G-t pedig adott karakterre lépésre.

A program 9 blokk hosszú programfájlokba menti ki a karakterkészletet, így e kimentett sorozat bármilyen programhoz használható, feltéve, hogy a 2 k-tól 4 k-ig terjedő területet a felhasználói program nem használja.

A program a következő részekből áll:

- 100—117 A képernyő kirajzolása és a kezdő adatok beállítása
- 119—127 A karakter képének kirajzolása, várakozás egy gomb lenyomására
- 128—159 Karakterkészlet-tervező funkciók
- 160—173 Karakterkészlet-kimentő rutin
- 180—193 Betöltő rutin
- 200—204 A lemezegység hibacsatornáját lekérdező rutin
- 210—220 Készletben ugró rutin
- 1000— Szabad terület esetleges bővítések számára

A program beírása vagy betöltése előtt ne felejtsük el beírni a következőt:
POKE 44,16:POKE 43,1:POKE 4096,0:NEW

OLÁH GERGELY

```
100 POKE650,128:PRINT"■■■":POKE53280,0:P
OKE53281,9
101 PRINT"■■+++++++++0  ■■*■      PONT KIR
AKASA."
102 PRINT"■+++++++++1  ■■DEL■   PONT TORL
ESE."
103 PRINT"■+++++++++2  ■■CRSR■  KURZOR MO
ZGATASA."
104 PRINT"■+++++++++3  ■■CLR■   KARAKTER
TORLESE."
105 PRINT"■+++++++++4  ■■HOME■  KURZOR FE
L. BALRA."
106 PRINT"■+++++++++5  ■■+■     KOVETKEZO
 KARAKTER."
107 PRINT"■+++++++++6  ■■-■     ELOZO KAR
AKTER."
108 PRINT"■+++++++++7  ■■S■     LEMEZRE M
ENTES."
109 PRINT"■76543210   ■■L■     BETOLTES
LEMEZROL."
110 PRINT"■             ■■G■     UGRAS VAL
AHOVA."
111 PRINT"■■■VARJON !!!■"
112 POKE56334,PEEK(56334)AND254:POKE1,PE
EK(1)AND251:FORI=0TO2047
113 POKE2048+I,PEEK(53248+I):NEXT:POKE1,
PEEK(1)OR4:POKE56334,PEEK(56334)OR1
114 POKE53272,(PEEK(53272)AND240)+2
115 PRINT"■KOD                       "
116 PRINT"KARAKTER:"
117 C=1
119 PRINT"■■■■■■■■■■■■■■■■■■■■";C
120 FORI=0TO7:A=PEEK(2048+(C*8)+I):FORJ=
7TO0STEP-1:POKE1024+40*I+(7-J),43
121 IF(A AND 2↑J)<>0THENPOKE1024+40*I+(7
-J),171
122 NEXT:NEXT
123 X=7:Y=0
124 POKE1024+(40*12)+11,C
125 POKE1024+9+(40*Y),31:POKE1024+(40*9)
+(7-X),30
126 GETA$:IFA$=""THEN126
127 POKE1024+9+(40*Y),32:POKE1024+(40*9)
+(7-X),32
128 IFA$<>"■"THEN131
129 Y=Y-1:IFY<0THENY=0
130 GOTO124
131 IFA$<>"■"THEN134
132 Y=Y+1:IFY>7THENY=7
133 GOTO124
134 IFA$<>"■"THEN137
135 X=X-1:IFX<0THENX=0
136 GOTO124
137 IFA$<>"■"THEN140
138 X=X+1:IFX>7THENX=7
139 GOTO124
140 IFA$<>"*"THEN143
141 POKE2048+(C*8)+Y,PEEK(2048+(C*8)+Y)O
R2↑X:POKE1024+(40*Y)+(7-X),171
142 GOTO124
143 IFA$<>CHR$(20)THEN146
144 POKE2048+(C*8)+Y,PEEK(2048+(C*8)+Y)A
ND(255-2↑X):POKE1024+(40*Y)+(7-X),43
145 GOTO124
146 IFA$<>"■"THEN149
147 FORI=0TO7:POKE2048+(C*8)+I,0:NEXT
148 GOTO119
149 IFA$<>"■"THEN152
150 X=7:Y=0
151 GOTO124
152 IFA$<>"+"THEN155
153 C=C+1:IFC=256THENC=255
154 GOTO119
155 IFA$<>"-"THEN158
156 C=C-1:IFC=-1THENC=0
157 GOTO119
158 IFA$<>"G"THEN160
159 GOSUB210:GOTO119
160 IFA$<>"S"THEN180
161 F$="":PRINT"■■■■■■■■■■■■■■■■FAJL NEV
E (16 K.):■ ■■";
162 GETA$:IFA$=""THEN162
163 IFASC(A$)>31ANDASC(A$)<96ANDLEN(F$)<
16THENPRINTA$;"■ ■■";:F$=F$+A$
164 IFA$=CHR$(20)ANDF$<>""THENPRINTA$;"■
■ ■■";:F$=LEFT$(F$,LEN(F$)-1)
165 IFA$=CHR$(13)THEN167
166 GOTO162
167 OPEN15,8,15,"I0":OPEN1,8,1,"@:"+F$+"
,P,W":A1=2048:A2=4095
168 GOSUB200
169 PRINT#1,CHR$(A1-INT(A1/256)*256);:GO
SUB200
170 PRINT#1,CHR$(INT(A1/256));:GOSUB200
171 FORI=A1TOA2:PRINT#1,CHR$(PEEK(I));:N
EXT:A=1:CLOSE1:GOSUB200:CLOSE15
172 PRINT"■■■■■■■■■■■■■■■■■■■
                  "
173 GOTO119
180 IFA$<>"L"THEN210
181 F$="":PRINT"■■■■■■■■■■■■■■■■FAJL NEV
E (16 K.):■ ■■";
182 GETA$:IFA$=""THEN182
183 IFASC(A$)>31ANDASC(A$)<96ANDLEN(F$)<
16THENPRINTA$;"■ ■■";:F$=F$+A$
184 IFA$=CHR$(20)ANDF$<>""THENPRINTA$;"■
■ ■■";:F$=LEFT$(F$,LEN(F$)-1)
185 IFA$=CHR$(13)THEN187
186 GOTO182
187 OPEN15,8,15,"I0":OPEN1,8,0,"0:"+F$+"
,P,R":A1=2048:A2=4095
188 GOSUB200
189 GET#1,A$:GOSUB200
190 GET#1,A$:GOSUB200
191 FORI=A1TOA2:GET#1,A$:POKEI,ASC(A$+"0
"):NEXT:CLOSE1:GOSUB200:CLOSE15
192 PRINT"■■■■■■■■■■■■■■■■■■■
                  "
193 GOTO119
200 INPUT#15,E1,E2$,E3,E4:IFE1=0THENRETU
RN
201 PRINT"■■■■■■■■■■■■■■■■■■■■■■■";E1;E2$;E3
;E4
202 GETA$:IFA$=""THEN202
203 PRINT"■■■■■■■■■■■■■■■■■■■■■■■
                  "
204 RETURN
210 IFA$<>"G"THEN1000
211 F$="":PRINT"■■■■■■■■■■■■■■■■■■HOVA (3
K.):■ ■■";
212 GETA$:IFA$=""THEN212
213 IFASC(A$)>47ANDASC(A$)<58ANDLEN(F$)<
3THENPRINTA$;"■ ■■";:F$=F$+A$
214 IFA$=CHR$(20)ANDF$<>""THENPRINTA$;"■
■ ■■";:F$=LEFT$(F$,LEN(F$)-1)
215 IFA$=CHR$(13)THEN217
216 GOTO212
217 C=VAL(F$):IFC<0THENC=0
218 IFC>255THENC=255
219 PRINT"■■■■■■■■■■■■■■■■■■■
                  "
220 RETURN
1000 RUN
```

Mit tud a PROLOG?

Az alábbiakban egy olyan programozási nyelvről lesz szó, amelynek filozófiája alapvetően eltér a köziesmert programozási nyelvektől.

A hagyományos programozási nyelvekben (FORTRAN, BASIC, FORTH, PASCAL, Ada stb.) a géppel való kapcsolat egyoldalú és egysíkú; a felhasználó mindig felszólító módban érintkezik a géppel: tedd ide, tedd oda, add össze, szorozd meg. E nyelvek ilyen, ún. imperatív vagy parancsosztó jellegével szemben a PROLOG egy egészen más, békésebb, erőszakmentes programozási stílust képvisel. A hagyományos nyelvek „parancs", „utasítás", „eljárás" stb. fogalmai helyett „tény", „állítás", „tudás" fogalmakban gondolkodik, és egy feladat megoldása a feladatot végrehajtó algoritmus elindítása helyett a feladatot leíró tudás bizonyos elemeire való rákérdezést jelent.

A PROLOG története

A mesterséges intelligencia kutatói általában az emberi gondolkodás egyik modelljének alapjául a matematikai logika tudományát tekintik. Ez irányú kutatások melléktermékeként fejlesztették ki a franciaországi Marseille-ben az első PROLOG interpretert, mint a matematikai logika egyik résznyelvében felírt matematikai tételek bizonyítására alkalmas eszközt. A francia kutatók ezt az első PROLOG interpretert a (francia) természetes nyelv feldolgozására (értelmezésére, fordítására) akarták használni.

Az eredeti interpreter egy példánya 1973-ban átkerült Angliába, az edinburgh-i egyetemre, ahonnan 1974-ben eljutott Magyarországra. Mivel a magyar kutatók ezt az interpretert nem tudták a hazai gépeken felélesztení, úgy döntöttek: írnak egy újat. Így jött létre az első magyar PROLOG interpreter 1977—78-ban. Eközben Edinburgh-ben elkészítették az első PROLOG fordítóprogramot is.

1981-ben lényeges változást hozott az, hogy a japánok bejelentették az 5. generációs számítógépek, vagyis az emberrel összehasonlítható intelligenciájú gépek fejlesztését. Noha a fejlesztés rendkívül hosszú távú célokat tűzött ki, azt már az elején eldöntötték, hogy ilyen színvonalú feladatok megfogalmazására csakis a PROLOG nyelv alkalmas. Elképzeléseik szerint az 5. generációs gépek a PROLOG-ot „helyből" fogják beszélni, úgy, mintha gépi kódjuk lenne.

Ezek az események nagy lökést adtak a nyelv fejlődésének. A világon sorra jelentek meg az újabb és újabb PROLOG rendszerek. Ezek közül most csak kettőt szeretnénk kiemelni.

Az egyik a hazai fejlesztésű **MPROLOG.** Nevében az M a modularitásra utal. A rendszer fő előnye az, hogy — éppen a modularitásból kifolyólag — rendkívül kifinomult eszközöket ad a felhasználó kezébe, olyanokat, amelyekkel több ember által megírt, igen nagy méretű és összetett programok is könnyen kezelhetőek. A rendszer külön jellegzetessége a T/TS PROLOG kiegészítés, amely az MPROLOG-ot párhuzamosan megoldandó szimulációs feladatok megfogalmazására is alkalmassá teszi.

A másik az angliai fejlesztésű **micro-prolog,** amely jóval szűkebb felhasználói lehetőségeket nyújt, viszont elsősorban mikroszámítógépeken terjedt el. A legkisebb micro-prolog-változat a mindössze 48 kilobájtos Sinclair ZX-Spectrum számítógépen fut, és példányai Magyarországon is megtalálhatók. A micro-prolog a Logic Programming Associates, a Sinclair ZX-Spectrum a Sinclair Research cégek védjegye.

Minthogy az MPROLOG a világon legelterjedtebb és szabványként elfogadott PROLOG nyelvtan ajánlásait követi, és a cikk írója is a magyar fejlesztőcsoport egyik képviselője, ezért a cikkben felhozott programpéldák az MPROLOG nyelvhez kötődnek.

A PROLOG elemei

Egy PROLOG program elindítása — mint már említettük — az előzőleg felvitt ismeretbázison alapuló kérdések feltevését jelenti, futása pedig a kérdésekre való válaszadás, illetve a kérdések, mint új állítások bebizonyítása.

Nézzük meg, hogy milyen eszközökkel adható meg az ismeretbázis. A PROLOG-világ két különböző osztályból áll:

— a tulajdonságok osztályából, amelyeket a programban különböző nevekkel (pl. szép, piros stb.) azonosítunk, és

— a dolgok, tárgyak osztályából, amelyek szintén lehetnek nevek, de lehetnek számok, vagy egyéb speciális szimbólumok is. Külön jelentősége van az ún. „nem definiált tárgyak" osztályának, amelyeket megkülönböztetésképpen a közönséges nevektől **nagy kezdőbetűkkel** írunk, ellentétben a többi nevekkel, amelyeknek mindig kisbetűvel kell kezdődniük.

Ha egy tulajdonságnevet nulla vagy több objektummal, mint paraméterekkel képzeletben összekapcsolunk, akkor egy **állításelemet** kapunk. Az állításelem tulajdonképpen csak egy kifejezési forma, amelynek önmagában nincs külön értelme.

Az állításelemet írásban úgy jelöljük, hogy a tulajdonság neve után zárójelbe téve és vesszőkkel elválasztva leírjuk a paraméterobjektumokat. Például a

szerelmes(csokonai,lilla)
kék(tenger)
zöld(Valami)

állításelemekben a „szerelmes", „kék", „zöld" tulajdonságneveket kapcsoltuk össze a „csokonai", „lilla", „tenger", objektumokkal és a „Valami" nem definiált objektummal.

Ha egy állításelemnek azt az értelmezést tulajdonítjuk, hogy az őt létrehozó tulajdonság a vele összekapcsolt objektumokra **mindig igaz,** akkor elemi állítást vagy **tényállítást** kapunk.

Az egyetlen állításelemből álló feltételt egyszerű feltételnek nevezzük. Helyes feltételekből a VAGY és az ÉS logikai műveletek segítségével képezhetünk újabb, **összetett feltételeket.** Ha egy feltételt és egy állításelemet mint következményt a HA—AKKOR logikai művelet segítségével kapcsolunk össze, akkor összetett állításokat vagy következtetések leírására szolgáló **szabályokat** kapunk. Az azonos nevű és paraméterszámú szabályokat együttesen **definíciónak,** az egy definíciót felépítő állításokat, és az egy állításban VAGY kapcsolóval összekapcsolt részfeltételeket **változatoknak** vagy **alternatívák**nak nevezzük. Egy szabály feltételrészében szereplő elemi feltételeket **hivatkozás**nak vagy röviden **hívás**nak nevezzük.

A tényállításokat úgy írjuk le, hogy vesszük az őket felépítő állításelemet, a szabályokat pedig, hogy a következményt és a feltételeket a ":—" következtetést jelző szimbólummal választjuk el, majd mindkettőt egy "." (pont) szimbólummal zárjuk le. A feltételek megfogalmazásánál vagy csak a megfelelő állításelemet vesszük (egyszerű feltételek esetén), vagy helyes feltételeket a "," (vessző) ÉS szimbólummal, illetve ";" (pontosvessző) VAGY szimbólummal elválasztva hozhatunk létre összetett feltételt. Az egyértelműség biztosítása végett megengedett a tetszőleges mélységű zárójelezés is. Például:

zöld(mező)
mellékbolygója(nap,merkur)
vizes a járda:—esik az eső;locsoltak.

A tényállítások és a következtetési szabá-

lyok szolgálnak logikai programunk építőköveiül. Egy PROLOG program megírása a program logikai magját alkotó tényállítások és szabályok megírását jelenti. A teljes PROLOG programban nyer igazi értelmet a változók használata: ha a programot — az algebrai egyenletrendszerekhez hasonlóan — **logikai állításrendszerként** fogjuk fel, akkor a változók jelentése teljesen megegyezik az algebrai egyenletrendszerekben használt változók értelmezésével; míg ott az egyenlet megoldásának célja a változók az egyenlet által megfogalmazott megkötéseket kielégítő értékeinek megkeresése, addig a PROLOG program futtatásának célja a logikai változók azon értékeinek megkeresése, amelyek felvétele esetén a célállítás a programból logikailag levezethető.

Megjegyezzük, hogy a PROLOG nyelvdefiníció szabályainak értelmében az ugyanolyan nevű változószimbólumok csak egyetlen (alap- vagy összetett) állításon belül tekintendők azonosnak, és ha egy változó a program futása során értéket kap, akkor az azonos változók is egyidejűleg felveszik ugyanazt az értéket. Külön állításban előforduló azonos nevű változók értéktartalma egymástól teljesen független. Egy PROLOG program végrehajtásának egy pillanatában azokat a változókat, amelyekhez a rendszer már talált behelyettesítési értéket, a talált értékkel **lekötött,** a többit **lekötetlen** vagy **szabad** változóknak nevezzük.

A már megtervezett és megírt PROLOG programban az állításelemek felhasználásával tehetünk fel kérdéseket úgy, hogy őket ÉS kapcsolóval összekapcsoljuk. Az ilyen összekapcsolt kérdéseket **célsorozat**nak nevezzük. Egy ilyen kérdés eredményeképpen a rendszer megpróbálja bebizonyítani, hogy az egyes állításelemek **logikailag egyszerre levezethetők-e** az ismeretbázis állításaiból.

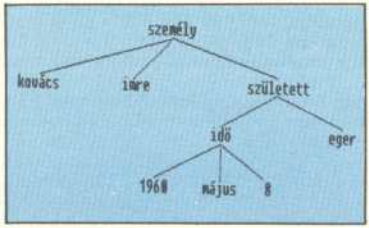
Noha más logikai műveletek segítségével is, de az adott műveletekkel is másféle szerkezeti állítások felépítése is elképzelhető, a PROLOG-ban alkalmazott megszorítások, vagyis hogy egy állítás következményoldala mindig egy egyszerű állításelem, valamint hogy a feltételoldalon csak az ÉS és VAGY műveleteket alkalmazhatjuk, biztosítják azt, hogy egy ilyen bizonyítás a gép által elvégezhető, és a programozó által áttekinthető, jól követhető legyen.

Az esetek legnagyobb részében az objektumok eddig elmondott osztályozása nem elegendő a megoldandó feladatok logikai alapú megfogalmazásához. Gyakran szükséges az objektumok szerkezetének finomabb ábrázolása, és hogy legyen lehetőség az objektumok egyes részeire is állításokat kimondani. Ezért az elmondott, ún. **egyszerű objektumok** fogalmat ki kell bővíteni, vagyis eszközöket kell adni **összetett objektumok** kezeléséhez is. A PROLOG-világban összetett objektumot képezhetünk úgy, hogy nulla vagy több egyszerű, vagy akár összetett objektumot egy felépítő (konstruktor) szimbólum segítségével egymással összekapcsolunk. Fontos megszorítás, hogy a felépítő szimbólum csakis egy egyszerű név lehet, amelyet az objektumra való bármilyen hivatkozás esetén ismerni kell.

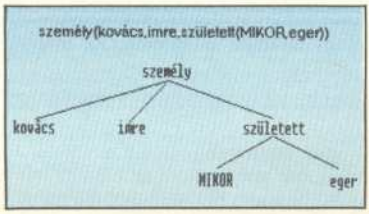
Az ily módon képzett objektumok olyan fákkal ábrázolhatók, amelyek csúcspontja a felépítő szimbólum, és amelyet a csúcspontból kiinduló ágak a paramétereket reprezentáló részfákkal kötnek össze. Természetesen az így képezhető faobjektumok is tartalmazhatnak változókat, értelemszerűen valamelyik levélelemük helyén. Egy változót is tartalmazó fa olyan félig meghatározott objektumot jelöl, amelynek a gyökér felé eső része teljesen meghatározott, és a teljes fa úgy válhat meghatározottabbá, hogy a változó helyére a programfutás során behelyettesítődhet egy olyan — változót tartalmazó vagy nem tartalmazó — objektum, ami kielégíti a program és a célállítás által meghatározott logikai feltételeket.

Megjegyezzük, hogy nem magától értetődő programtervezői munka egy adott feladat követelményeihez legjobban illeszkedő objektumszerkezet megtervezése.

Az összetett objektumok írásmódja: a konstruktorszimbólum után zárójelek között, vesszőkkel elválasztva leírjuk az összetett szimbólumokat. Például a
személy(kovács,imre,született(idő(1960,május,8),eger))
a következő faobjektumot jelenti:



A nem teljesen meghatározott objektumokra példa a következő, ahol a nem teljes meghatározottság az előzőhöz viszonyítva a születési időt jelző részben jelentkezik:



Az MPROLOG nyelv lehetőséget ad bizonyos szimbólumok ún. **operátor**ként való deklarációjára. Az operátorszimbólumokat egy vagy két összetevős összetett objektumok leírásánál használjuk; ha egy operátort **közbeékelődő**nek (infix) nyilvánítunk, akkor amennyiben a szimbólum kétösszetevős kifejezést épít fel, úgy a megszokott zárójelek jelölésmód helyett a szimbólumot a zárójelek nélkül, a kétösszetevő objektummal közrefogva is írhatjuk, így is ugyanazt az összetett kifejezést jelenti. Azért, hogy a többszörösen összetett kifejezések használata is egyértelmű legyen, a

```
module árpád_ház.

/*$eject*/
body.

operator(apja,lr,0).
operator(nagyapja,lr,0).
operator(király,st,0).

árpád apja álmos.
solt apja árpád.
taksony apja solt.
mihály apja taksony.
géza apja taksony.
vazul apja mihály.
1.endre apja vazul.
1.béla apja vazul.
salamon apja 1.endre.
1.géza apja 1.béla.
1.László apja 1.béla.
1.kálmán apja 1.géza.
1.istván apja géza.
imre apja 1.istván.

1.endre király.
1.béla király.
1.géza király.
1.László király.
1.kálmán király.
salamon király.
1.istván király.

UNOKA nagyapja NAGYPAPA :-
    PAPA apja NAGYPAPA, UNOKA apja PAPA

:- VALAKI nagyapja taksony, VALAKI király.

endmod /* árpád_ház */.
```

deklarációban jelezni kell a zárójelezés irányát és az új operátor prioritását is.

Ha egy szimbólumot **elő-** (prefix) vagy **utóképző** (suffix) operátornak deklarálunk, akkor egyösszetevős kifejezéseket zárójelezés nélkül, a felépítő szimbólumot az egyetlen összetevő elé, illetve után írva is kifejezhetjük magunkat.

Megjegyezzük, hogy egy MPROLOG állítás tulajdonképpen egy összetett kifejezés formájú, hiszen az egyes állításelemek felépítése megegyezik az összetett kifejezések ábrázolásmódjával, és a HA—AKKOR, a VAGY és az ÉS szimbólumok (":—" ";" ",") szintén mint alapoperátorok vannak deklarálva.

Az operátorok használata jól megfigyelhető a fenti programban, amely az Árpád-házi királyok egy részének családfáját írja le.

Végrehajtási algoritmus

Az algebrai egyenletekhez hasonlóan a PROLOG nyelven felírt logikai állításrendszereknek is van megoldási módszerük. Tegyük fel, hogy adott egy PROLOG állításrendszer és adott egy célállítás-sorozat, amelyet bizonyítani kívánunk. A megoldási módszer lényege: módszert adunk egy célsorozathoz olyan újabb célsorozatok keresésére, hogy ha az új célsorozat bizonyíthatóan teljesül, akkor az eredeti célsorozat-

nak is teljesülnie kell. Ha ilyen műveletek alkalmazása során a mindig teljesülő, egyetlen célállításelemet sem tartalmazó **üres** célsorozathoz jutunk, akkor ebből következően a kiindulási célsorozatnak is teljesülnie kell.

A módszer leglényegesebb lépése a szabályok „visszafelé" alkalmazása. Definíció szerint ugyanis egy szabály akkor alkalmazható visszafelé, ha a következményrésze illeszkedik a hívásra, vagyis ha:

— a hívás neve és paramétereinek száma megegyezik a szabály következményrész nevével és paramétereinek számával,

— a paramétereik páronként vagy megegyeznek, vagy a bennük szereplő változókba behelyettesíthetők objektumok úgy, hogy azzal megegyezővé tehetők.

Ezt az illesztés-behelyettesítési lépést a PROLOG nyelvben **egyesítésnek** nevezik. Az egyesítés pontosabb definíciója: minden objektum egyesíthető önmagával; egy változó minden objektummal egyesíthető úgy, hogy a változó felveszi a másik objektumot értékül; egy összetett objektum akkor illeszkedik egy másik összetett objektumhoz, ha a nevük és a paraméterszámuk megegyezik, és a paramétereik páronként egyesíthetők; más esetben a két objektum nem egyesíthető.

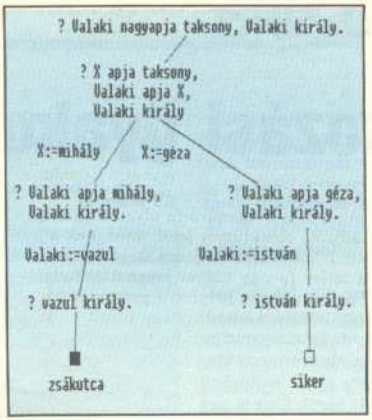
Ezek figyelembevételével a végrehajtási módszer a következő. Ha egy következtetési szabályt „visszafelé" alkalmazunk, azaz a célsorozat valamelyik elemére illeszkedő következményszerű szabályt találunk, és egyesítés után a célállításelemet kicseréljük a szabály feltételrészére, akkor kapunk egy új célsorozatot, ami rendszerint hosszabb, de egyszerűbb állításelemekből áll, mint az eredeti. Ha illeszthető tényállítás is, az felfogható üres feltételrészű szabálynak is. Ez azt jelenti, hogy az adott állításelem közvetlenül teljesül, és ilyenkor ez az állításelem a még bizonyítandó célállítás-sorozatból elhagyható.

Ehhez az általános módszerhez a PROLOG nyelvdefiníció még a következő megszorításokat teszi hozzá:

— Mindig az aktuális célsorozat első elemével kell az illesztést megkísérelni.

— Ha az aktuális célsorozat első elemére több szabály vagy tényállítás is illeszkedik, akkor azt mondjuk, hogy **választási ponthoz** érkeztünk. Ilyenkor mindig a programszöveg szerinti legelső választási lehetőség szerint kell a végrehajtást folytatni, és a többi ág végigjárására csak később kerülhet sor.

— Ha az aktuális célsorozat első elemére egyetlen szabály vagy tényállítás sem illeszkedik, akkor azt mondjuk, hogy a bizonyítás **zsákutcába** jutott. Ilyenkor meg kell kísérelni a legutoljára elhagyott választási pontból a következő választási lehetőség szerint folytatni a végrehajtást. Ezt a műveletet **visszalépésnek** (backtracking) nevezzük. Fontos megjegyezni, hogy a visszalépés során a program teljesen visszalép korábbi állapotába, így például az időközben lekötött változók ismét felszabadulnak. Amennyiben az adott választási pontnál már az összes választási lehetőséget kimerítettük, akkor egy korábbi választási pontnál kell a további végrehajtást folytatni, és ha már a legkorábbi választási pontnál sincs szabad lehetőség, akkor azt mondjuk, hogy a bizonyítás **kudarccal** végződött, az eredeti célsorozat az adott ismeretbázis feltételei mellett nem teljesül.



Levezetési fa

PROLOG programok végrehajtása jól szemléltethető az ún. levezetési fa segítségével. A fa gyökere a kiindulási célsorozat, és az egyes csúcsai a levezetés során előálló újabb célsorozatok. A levezetési fa éleire a megfelelő egyesítési lépés által létrehozott változó megkötéseket szokás írni. Példaképpen tekintsük az Árpád-ház programhoz feltett

? Valaki nagyapja taksony, Valaki király.

kérdés alapján előálló levezetési fát.

Beépített állítások

Minden PROLOG rendszerben vannak olyan állítások, amelyeket a felhasználónak nem szükséges definiálnia, amelyeket a rendszer helyből ismer. Az ilyen állítások rendszerint valamilyen alapadattípuson értelmezett relációt valósítanak meg, bizonyos részük a rendszer egészét érintő (globális) mellékhatással is rendelkezik. Ezek az állítások általában érzékenyek arra is, hogy meghívásuk pillanatában mely paraméterük van lekötött, illetve szabad állapotban, és amennyiben a programozó nem ügyel ezekre a megszorításokra, akkor a rendszer hibaállapotba kerül, vagy egyéb nem várt módon reagál.

Az MPROLOG rendszerben megvalósított beépített állítások főbb csoportjai:

— Periféria- és fájlkezelő eljárások; betűk, jelek, szavak, illetve MPROLOG kifejezések kiírása, beolvasása, lemez, nyomtató, képernyő kezelése.

— Aritmetikai relációk: egész- és lebegőpontos tizedes számokból az alapműveletek segítségével felépülő számtani kifejezések értékének kiszámítása, összehasonlítások.

— Kifejezés- és füzérkezelés: MPROLOG kifejezések futásidejű felépítése, illetve lebontása, szövegdaraboló, összefűző, összehasonlító eljárások.

— Végrehajtást befolyásoló eljárások.

— Műveletek a programok szerkezetén: programmodul, moduleljárás, relációk, állítások törlése, új állítások felvétele.

— Hibaállapot-lekezelő eljárások.

A "!" (out) beépített eljárás

A "!" olyan beépített eljárás, amely mellékhatása miatt egyáltalán nem értelmezhető logikai állításként, és működése csak a PROLOG végrehajtási mechanizmus ismeretében érthető meg. Használatával a PROLOG nyelv holmi PASCAL vagy Ada színvonalára „fokozható le", amennyiben a nyelv egyik fő erejét, a logikai működésmódot biztosító visszalépést tilthatjuk le vele. Használatát a levezetési fa ismeretében a legkönnyebb értelmezni.

Képzeljük el, hogy a levezetési fán minden szabályegyesítést jelképező élhez megjegyezzük a felhasznált szabály valamilyen azonosítóját is. Nevezzük ezt a pontot szabálybelépési pontnak. Amikor a "!" egyesítése kerül sorra, azaz amikor az aktuális célsorozatnak éppen egy "!" az első eleme, akkor általában már messze járunk a rá hivatkozó szabály belépési pontjától, és rendszerint a közöttük fekvő úton több nyitott elágazás, visszalépési pont is található. A "!" eljárás hatására a rá hivatkozó szabály belépési pontjáig az összes visszalépési pont megszűnik, mert a "!" mintegy „levágja" a levezetési fa nyitott ágait.

Program-végrehajtási szempontból a "!" hatására saját definíciójának további alternatíváira és a feltételsorban őt megelőző hívások által nyitva hagyott visszalépési pontokra már sohasem kerülhet a vezérlés.

Összefoglalás

A PROLOG nyelv megjelenése hatalmas előrelépést jelentett elődeihez viszonyítva. A nyelv egzakt értelmezése következtében matematikai módszerekkel jól kezelhető és leírható. Ugyanezen okokból, valamint a programok és adatok egységes ábrázolásmódja miatt a nyelv maga is gépi eszközökkel is rendkívül könnyen, rugalmasan kezelhető.

A nyelvben alkalmazott főbb újszerű technikák, a visszalépés, a mintaillesztés, a futásidejű programkezelés stb. a nyelvnek olyan erőt adnak, amelyet elsősorban olyan feladatokhoz ajánlott alkalmazni, ahol különleges bonyolultságú döntéseket kell végezni, vagy logikai következtetéseket kell levonni. A nyelv különösen alkalmas egyedi igényeket kielégítő alkalmazói programok gyors előállítására. Főbb alkalmazási területei: a természetes nyelvek feldolgozása, adatbázisok természetes nyelvű lekérdezése, a következtető rendszerek, például szakértői rendszerek következtető alrendszere és a számítógéppel segített tervezés (építészet, gyógyszertervezés, áramkörtervezés stb.).

KILIÁN IMRE

Z80 programozási gyakorlatok 6.

Ahogy az előző részben megígértem, most megnézzük, hogy pontosan mi is az az interrupt (magyarul: megszakítás), és assembly programozás során milyen feladatok adódhatnak ebben a témakörben.

A Z80 processzornak két megszakításbemenete van. Ha az NMI (nem maszkolható megszakítás) bemenetre érkezik megszakításkérés, akkor a processzor a következő beolvasott utasítást figyelmen kívül hagyja. Először a programszámlálónak, amely már az új utasításra mutat, a felső, majd alsó bájtja a verembe kerül, miközben az SP kétszer csökken eggyel, majd a PC-be 66H kerül. Szokás úgy fogalmazni, hogy a processzor egy újraindítást hajt végre a 66H címre. Ez a megszakításkérés nem maszkolható, azaz szoftver úton nem lehet tiltani az elfogadását.

A processzor másik megszakításbemenete az INT láb. Az erre a bemenetre érkező megszakításkérés elfogadása letiltható a DI utasítással, és engedélyezhető az EI utasítással. A valóságban két megszakításengedélyező bit van a központi egységben, szokásos jelölésük IFF1 és IFF2. Ha IFF1 értéke 1, akkor engedélyezve van a megszakítás, ha nulla, akkor a megszakításkérésről nem vesz tudomást a processzor. A DI utasítás mindkét bitbe nullát tesz, az EI pedig 1-et. Ez utóbbi végrehajtása után még a következő utasítást is végrehajtja a processzor, és csak ezután fogad el egy esetleges megszakításkérést. Ennek az az oka, hogy a megszakításrutin teljes lefutásáig általában nem fogadható el újabb megszakítás, márpedig az EI után kell hogy legyen még legalább egy RET utasítás.

Egy maszkolható megszakításkérés elfogadása után a processzor mindkét bitet törli, ezzel a további megszakításkérések elfogadását tiltja. IFF2 feladata az, hogy nem maszkolható megszakítás elfogadásakor megőrizze IFF1 tartalmát. Ilyenkor IFF1 értéke IFF2-be, IFF1-be pedig nulla kerül. Az NMI rutin végrehajtása végén a RETN utasítás hatására IFF1-be visszakerül IFF2-ből az eredeti érték, így ha tiltva volt, akkor tiltva, ha engedélyezve volt, akkor engedélyezve marad a maszkolható megszakítás az NMI rutin lefutása után is.

Foglalkozzunk a továbbiakban a maszkolható megszakításokkal. A Z80-nak három megszakítás üzemmódja van.

6.1 program

```
;  6.1 program

        ORG   65000       ;interrupt
SETINT  DI                ;
        LD    A,9         ; beállítása
        LD    I,A         ;I=9
        IM    2           ; kettes interrupt mód
        EI                ;
        RET               ;
        ORG   65129       ;a kiszámított cím
ORA     PUSH  AF          ;regiszterek
        LD    HL,CLOCK    ;HL végig az óra megfelelő
                          ;bájtjára mutat
        LD    A,50        ;1 sec=50*20 ms
        CALL  STEP        ;
        LD    A,60        ;1 min=60 sec
        CALL  STEP        ;(akku megmarad)
        CALL  STEP        ;1 h=60 min
        LD    A,24        ;1 d=24 h
        CALL  STEP        ;
RETIT   POP   HL          ;regiszterek
        POP   AF          ;  vissza
        EI                ;engedélyezni kell!
        RET               ;az interruptot
STEP    INC   (HL)        ;megfelelő bájt növelése
        CP    (HL)        ;elérte-e a limitet ?
        JR    NZ,RETURN   ;nem ------------------------+
        LD    (HL),0      ;törlés                    |
        INC   HL          ;mutató növelése           |
        RET               ;normális visszatérés      |
RETURN  POP   HL          ;visszatérési cím elhagyása <-+
        JR    RETIT       ;vissza az interruptból
        END               ;
```

6.2 program

```
        ORG   65129       ;interrupt cím
MAX     EQU   10          ;a buffer max mérete
MASINT  EQU   $           ;
        PUSH  AF          ;regiszterek
        PUSH  BC          ;    elmentése
        PUSH  DE          ;
        PUSH  HL          ;
        LD    HL,CLOCK    ;az óra
        DEC   (HL)        ;  csökkentése
        LD    HL,FLAG     ;van e
        BIT   1,(HL)      ; billentyűzet
        JR    Z,KHIT      ;        kérés
        CALL  RDB         ;puffer olvasás
        OR    A           ; ha
        JR    Z,KHIT      ;  üres a puffer
        LD    (LAST),A    ;a billentyű érték
        LD    HL,FLAG     ;érvényes billentyű
        RES   1,(HL)      ;              jelzése
KHIT    CALL  READ        ;a billentyűzet leolvasása
        OR    A           ;
        JR    NZ,NOKEY    ;nincs billentyű
        LD    HL,FLAG     ;
        BIT   0,(HL)      ;első ?
        JR    NZ,REPK     ;ismételtnél ugrás
        SET   0,(HL)      ;volt billentyű beállítása
        CALL  WRTB        ;beírás a pufferbe
        LD    A,(ERT1)    ;az óra beállítása
        JR    STEC        ;
REPK    LD    B,A         ;ismételt billentyű
        LD    A,(CLOCK)   ;az óra
        OR    A           ;  nulla-e ?
        JR    NZ,RETURN   ;ha nem akkor visszatérés
        LD    A,B         ;billentyű érték vissza A-ba
        CALL  WRTB        ;beírás a pufferbe
        LD    A,(ERT2)    ;ismételt várakozás
STEC    LD    (CLOCK),A   ;óra beállítása
        JR    RETURN      ;visszatérés
NOKEY   LD    HL,FLAG     ;nem volt
        RES   0,(HL)      ;lenyomott billentyű
RETURN  POP   HL          ;regiszterek
        POP   DE          ;    vissza
        POP   BC          ;
        POP   AF          ;
        EI                ;
        RET               ;
;olvasása a pufferből
RDB     LD    A,(BL)      ;A-ba a puffer hossza
        OR    A           ;zero bit állítása
        RET   Z           ;ha üres a puffer
        DEC   A           ;puffer hossza
        LD    (BL),A      ;  eggyel csökken
        LD    HL,(FD)     ;a kiolvasandó karakter
        LD    A,(HL)      ;      A-ba
        CALL  STEPHL      ;a mutató
        LD    (FD),HL     ;       léptetése
        RET               ;
;írás a pufferbe
WRTB    LD    B,A         ;karakter kód B-be
        LD    A,(BL)      ;a puffer hossza ?
        CP    MAX         ;tele
        RET   Z           ;  van a puffer
        INC   A           ;puffer hossza
        LD    (BL),A      ;  eggyel nő
        LD    HL,(FI)     ;a beírandó karakter
        LD    (HL),B      ;              a helyére
        CALL  STEPHL      ;a mutató
        LD    (FI),HL     ;       léptetése
        RET               ;
;HL-t a puffer következő elemére állítja
STEPHL  PUSH  HL          ;HL elmentése a vizsgálat miatt
        LD    DE,(BE)     ;DE=puffer vége
        OR    A           ;carry=0
        SBC   HL,DE       ;carry=1 ha HL<DE
        POP   HL          ;HL vissza
        INC   HL          ; normális léptetés
        RET   C           ;HL nem tul nagy
        LD    HL,(BS)     ;ha HL tul nagy akkor a puffer elejére
        RET               ;
;V A L T O Z O K
FD      DEFW  BUFFER      ;pufferből kiolvasandó karakter címe
FI      DEFW  BUFFER      ;pufferbeli első szabad hely
BS      DEFW  BUFFER      ;puffer kezdete
BE      DEFW  BUFFER+MAX  ;puffer vége
LAST    DEFS  1           ;a futó program számára
FLAG    DEFB  0           ;alsó két bit
CLOCK   DEFS  1           ;óra
BL      DEFB  0           ;a puffer pillanatnyi hossza
ERT1    DEFB  35          ;első várakozás
ERT2    DEFB  5           ;ismételt várakozás
;billentyűzet leolvasása
; (rendszerfüggő)
READ    EQU   $           ;billentyűzet leolvasása
        CALL  KYSCN       ;key scan ROM rutin (spectrum)
        LD    A,D         ;ha nincs
        AND   E           ;billentyű akkor
        INC   A           ;DE=0FFFFh
        RET   Z           ;ha nincs billentyű
        LD    D,0         ;DE:=a billentyű kódja (nem ASCII!)
        LD    HL,TBL      ;táblázat kezdete
        ADD   HL,DE       ;az ASCII kód címe
```

0 mód. Ebben az üzemmódban a megszakítást kérő eszköz bármilyen utasításkódot tehet az adatbuszra, a központi egység végrehajtja azt.

1. mód. A megszakításkérés elfogadásakor a központi egység egy RST 38H utasítást hajt végre.

2. mód. A megszakítást kérő eszköz által az adatbuszon küldött bájt alkotja egy cím alsó bájtját, az I regiszter a cím felső bájtját. Az így kapott címről veszi a processzor a megszakításrutin kezdőcímét.

Ezeket a módokat az IM (interrupt mode) utasításokkal lehet kiválasztani. Az első két mód elég egyszerű. Nézzünk meg egy konkrét példát a harmadik, azaz kettes üzemmód használatára.

Tegyük fel, hogy az I regiszterbe 9-et töltöttünk, a megszakításkérő eszköz pedig 255-öt küld az adatbuszon. Ekkor a cím 9*256+255=2559. Ha most ezen a címen 105, a következőn, a 2560-as címen 254 található, akkor a megszakításrutin kezdőcíme 254*256+105=65129.

Mire való a megszakítás? Nos, ez eléggé gépfüggő. Tegyük fel, hogy 20 ms-onként történik megszakításkérés. Ennek segítségével megvalósíthatunk egy belső órát a gépben, készíthetünk BREAK vizsgálatot, amellyel minden, a megszakítást le nem tiltó gépi kódú program megállítható, továbbá ismétlő billentyűzet-leolvasást csinálhatunk.

6.1 feladat

Valósítsunk meg a gépben 2-es megszakítás üzemmódban egy órát, amely az ötvened másodperceket, másodperceket, perceket és órákat számlálja!

Mivel 20 ms-onként, azaz ötvened másodpercenként kerül a vezérlés a megszakításrutinra, ezért ha minden megszakítás alkalmával egy bájtot növelünk, akkor az az ötvened másodperceket fogja mutatni. Amikor eléri az ötvenet, azaz letelt egy másodperc, akkor nullázni kell, és egy másik bájtot, a másodpercszámlálót kell növelni. Amikor az eléri a hatvanat, akkor a következővel kell hasonló módon eljárni, és így tovább.

A *6.1 program* két részből áll. Az első beállítja az I regisztert és az üzemmódot. Ez a rész bárhova kerülhet a tárba. A második rész viszont csak arra a bizonyos címre kerülhet, amelyet meghatároztunk. A mai példánkban a 65129-es címre. Ez megfelel a Spectrum ROM-jának és hardverjének. A bájtok egymás utáni növelését úgy oldották meg, hogy ha már nem kell növelni a következő bájtot, akkor a STEP szubrutin a visszatérési címet kiveszi a veremből, így nem minden CALL után tér vissza a program.

Nagyon fontos, hogy meg kell őrizni minden regiszter értékét, hiszen a főprogram nem tud gondoskodni azok elmentéséről.

A második feladat, a BREAK gomb vizsgálata rendkívül egyszerű. Tulajdonképpen azt kell tenni, hogy ha a gomb meg van nyomva, akkor nem tér vissza a program oda, ahol a megszakítás előtt

```
        LD    A,(HL)        ;A:=ASCII kód
        RET                 ;
;T A B L A Z A T
TBL     DEFM  'BHY65TGV'    ;
        DEFM  'NJU74RFC'    ;
        DEFM  'MKI83EDX'    ;
        DEFB  SSH           ;symbol shift
        DEFM  'LO92WSZ'     ;
        DEFM  ' '           ;szóköz
        DEFB  CR            ;carrige return
        DEFM  'P01QA'       ;
        DEFB  CSH           ;caps shift
BUFFER  DEFS  MAX           ;puffer
KYSCN   EQU   28DH          ;spectrum ROM rutin
CSH     EQU   0FFH          ;
SSH     EQU   0FEH          ;
CR      EQU   0DH           ;
        END
```

volt, hanem elindítja a gép alaprendszerét. Arra kell ügyelni, hogy a BREAK gomb hosszan le lehet nyomva, és bár egy másodpercig tartó lenyomásnál a megszakításrutin ötvenszer fogja érzékelni, mégis, csak egyszer szabad rá a fent leírt módon reagálni.

6.2 feladat

Valósítsunk meg a gépben 2-es megszakítás üzemmóddal egy billentyűzetleolvasást, amely fogadja a billentyűt, amikor a felhasználó azt lenyomja, majd ha lenyomva tartja, akkor 0,7 másodperc múlva ismét fogadja, majd ezután minden 0,1 másodpercben mindaddig, amíg a felhasználó fel nem engedi a billentyűt. Ha az éppen futó programnak nincs szüksége bemenetre, akkor tárolja a billentyűértékeket.

Ha minden másodpercben 50 megszakítás történik, akkor minden megszakítás esetén csökkentve egy számlálót, egy belső órát kapunk, amelyet ha egy billentyű első érzékelésekor 35-re állítunk, akkor 0,7 másodperc múlva lesz nulla, ha 5-re, akkor 0,1 másodperc múlva. Azt, hogy egy billentyűt először érzékel-e a megszakítás, úgy tudjuk vizsgálni, hogy ha nincs billentyű lenyomva, akkor egy bitet, a programban a flag változó nullás bitjét, nullára állítjuk, és ha van lenyomott billentyű, akkor 1-re. Így ha ez a bit nulla, de van lenyomott billentyű, akkor azt először érzékeli a megszakításrutin. Ekkor a billentyű értékét eltesszük egy pufferbe, az órát pedig 35-re állítjuk. Ha a bit nem nulla és van lenyomott billentyű, akkor azt csak akkor tesszük be a pufferbe, ha az óra nulla, ellenkező esetben ugyanis nem járt le a 0,7 vagy 0,1 másodperces várakozás.

A megszakításrutin másik feladata az, hogy kiszolgálja az éppen futó programot, ha az jelzi, hogy szüksége van egy billentyűértékre. Ha a gépben futó programnak egy értékre van szüksége, akkor a flag változó egyes bitjét 1-re állítja. A megszakításrutin innen tudja, hogy a pufferből egy értéket át kell tennie a megfelelő helyre (LAST változó), majd amikor az áthelyezés megtörtént, akkor a bitet nullázza, jelezve, hogy érvényes adat van a változóban. A futó program számára ez azt jelenti, hogy SET utasítással 1-re állítja a bitet, majd addig vár, amíg a bit nem törlődik.

Nézzük meg részletesen a puffert! Két műveletet kell megvalósítanunk.

1. Egy bájt beírása a pufferbe,
2. Egy bájt kiolvasása a pufferből.

Mindezt úgy, hogy mindig azt a bájtot kell kiolvasnunk, amit legrégebben írtunk be. Az ilyen szervezésű tárakat, puffereket FIFO-nak (first in, first out=első be, első ki) hívják. Ellentétben a vermekkel, amelyekből azt vesszük ki először, amit utoljára tettünk be. Ezeket LIFO-nak (last in, first out=utolsó be, első ki) hívják.

Egy FIFO puffer megvalósítása kétféle lehet. Az egyik, hogy mindig ugyanoda helyezzük a beírandó bájtot, és a többi, pufferben lévő bájtot LDIR vagy LDDR utasítással arrébb helyezzük. Ez a megoldás lassú a felesleges adatmozgatás miatt.

Másik megoldásként két mutatót használva írjuk és olvassuk a puffert. Ennek a megoldásnak az a hátránya, hogy ekkor a puffer mozog a tárban, sőt egy rosszul megírt program esetén végigvándorol az egész táron, ami menthetetlenül a rendszer lefagyását okozza. Ennek elkerülésére módosítjuk a két mutató mozgatását, olyan módon, hogy nem növeljük, illetve csökkentjük őket még nélkül, hanem amikor bármelyik eléri a pufferre szánt terület végét, akkor a puffer elejére mutató értéket töltjük belé. Regisztrálni kell azt is, hogy van-e még hely a pufferben, mert így az egyik mutató utolérheti a másikat, illetve olvasáskor azt, hogy van-e a pufferben kiolvasható adat.

BASIC és gépi kód

Legutóbb a gépi kódú programokat mentő rutin működését néztük meg közelebbről, és szó volt egy meglepő felfedezésről. Most néhány újabb gépi kódú utasítás ismertetése után egy újabb — számomra kellemetlen — felfedezésről számolok be.

A számláló utasítások

Az ebbe a csoportba tartozó utasítások az indexregiszterek vagy valamelyik memóriabájt tartalmát növelik, illetve csökkentik eggyel. Nevük az angol increment (növekedés), illetve decrement (fogyás) szó első betűiből és indexregiszterekre vonatkozó utasítások esetén a regiszter nevéből állnak. Az INC, INX és INY a növelő, a DEC, DEX és DEY a csökkentő utasítások. Az A regiszter tartalmát sajnos nem lehet hasonló utasítással eggyel növelni vagy csökkenteni, de viszonylag egyszerűen megoldható.

A számláló utasítások az állapotregisztert az N és a Z bitjét állítják be, a korábban ismertetett módon. A C (átvitel) bit állapotát nem változtatják, tehát ennek a figyelése nem tájékoztat arról, hogy például növelésnél elértük-e a $FF után a $00 értéket. (A C bitről eddig még nem volt szó, az eddig ismertetett utasítások egyike sem változtatja meg az állapotát.)

A közvetett JMP utasítás

A JMP utasítás ismertetésekor csak az abszolút címzési módú változatról írtam. Nem kevésbé fontos a közvetett (indirekt) címzési módú JMP utasítás sem. Működése egy kicsit bonyolultabb az előbbiénél. Az utasítás operandusa egy tárcím, de az ugrás nem erre a címre történik. Az operandus által megadott helyen egy további kétbájtos cím információ található, a már többször említett alsó bájt/felső bájt sorrendben. Az ugrás erre a címre történik.

A közvetett JMP utasítás gépi kódja $6C, az assembly formájú leírásban úgy különböztetjük meg az egyszerű abszolút címzésűtől, hogy az operandust zárójelbe tesszük. Például: JMP ($0300).

Az újabb felfedezés

Legutóbb írtam arról a felfedezésemről, hogy kazettás tárolóról történő programbetöltés esetén a gépi kódú program egy egyszerű LOAD parancs hatására az eredeti helyére töltődött. Hozzátettem, hogy ilyesmiről eddig még sehol sem olvastam. Az újabb felfedezésem az, hogy olvashattam volna, csak egy kicsit jobban oda kellett volna figyelnem. A Commodore 64-es első felépítése című könyv 107. oldalán — melyet korábban többször is átolvastam — a következőt találtam: »LOAD „NÉV",1 — BASIC program relatív betöltése, ill. a 3-as típusú gépi kódú program abszolút betöltése«.

Ez magyarázatot ad a múltkori „csodálatos felfedezésemre". Elnézést kérek. A VC20-ra változatlanul a korábban leírtak érvényesek.

BARNA LÁSZLÓ

A *6.2 programban* WRTB szubrutin az akkumulátorban megadott bájtot írja be a pufferbe, ha van hely. Az RDB szubrutin egy bájtot olvas ki a pufferből, illetve nulla akkumulátortartalommal tér vissza, ha üres a puffer, ugyanis nulla ASCII kódú karakter nincs.

Az IT rutin először csökkenti az órát, majd megvizsgálja, hogy a futó program számára kell-e érvényes adat. Ha kell, akkor megpróbálja a pufferből kiolvasni, és ha a puffer nem üres, akkor a kiolvasott értéket elteszi a LAST változóba, majd törli a flag 1-es bitjét.

Ezután, illetve ha a puffer üres, vagy ha nem kellett LAST változót állítani, elkezdődik a rutin második része. A READ rutin leolvassa a billentyűzetet, és a lenyomott billentyű kódjával tér vissza. Ez a rész meglehetősen rendszerfüggő. Ha nincs lenyomott billentyű, akkor a visszaadott érték nulla, és ekkor a futás a NOKEY címkénél folytatódik. Ha van lenyomott billentyű, akkor megnézi, hogy az előző megszakításnál észlelt-e a program billentyűt, azaz új billentyű lenyomásáról van-e szó, vagy ismételni kell a billentyűt. A program elteszi az új billentyűértéket a pufferbe, jelzi a flag változó nullás bitjében, hogy volt billentyű ebben a megszakításban, majd a CLOCK változóba átmásolja ERT1 változó értékét, amely alaphelyzetben 35.

Ha nem új billentyűről van szó, akkor megvizsgálja, hogy az óra nulla-e. Ha nem, akkor a billentyűértéket figyelmen kívül kell hagyni. Ha nulla, akkor elteszi az értéket a pufferbe, az órát pedig ERT2-nek megfelelően állítja be. Ha nem volt billentyű lenyomva, akkor a flag változó nullás bitjének törlése után visszatér.

Az RDB (read buffer) szubrutin először megnézi a puffer hosszát (BL = buffer lenght), ugyanis ha ez nulla, akkor üres a puffer. Ha nem, akkor BL-t csökkenti, a FO (first out) által mutatott címről kivesz egy bájtot, majd lépteti FO-t a puffer következő elemére.

A felhasználói program kiszolgálását úgy is meg lehet oldani, hogy rendelkezésére bocsátunk egy szubrutint, amely a pufferből kiolvas egy értéket. Az út járható, de nagyon vigyázni kell, mert ekkor a megszakításrutin és az éppen futó program ugyanazon az adathalmazon végez változtatásokat.

Elképzelhető, hogy a puffer tele van, amikor olvasni akarunk belőle. FO-t léptetjük, BL-t csökkentjük, és kiolvasunk a pufferből egy kódot. Ha nem tiltjuk az olvasás idejére a megszakítást, megtörténhet, hogy FO léptetése és BL csökkentése után egy megszakítás a még ki nem olvasott bájtra teszi az új billentyűértéket, hiszen FO és BL értéke alapján ez a hely már szabad, és a megszakítás lefutása után ezt az értéket fogja a programunk kiolvasni.

A WRTB (write buffer) szubrutin is BL vizsgálatával kezdődik, de most azt nézzük, hogy nem telt-e meg a puffer. Ha nem, akkor BL-t növeljük, a FI (first in) által mutatott címre letesszük az értéket, és léptetjük FI-t.

Ennek a billentyűzetleolvasásnak van egy hátránya. Ha gyorsan gépelünk, és egy billentyűt úgy nyomunk le, hogy az előző felengedése óta még nem volt megszakítás, akkor az így lenyomott billentyűt a rutin ismételt billentyűnek tekinti.

Próbáljuk meg átírni a rutint úgy, hogy egy billentyűt akkor tekintsen újnak, ha az előző megszakításban nem volt billentyű lenyomva, vagy ha más billentyű volt lenyomva!

A megszakításrutin változói:

ERT1	1 bájt	billentyűismétlésnél az első várakozás hossza
ERT2	1 bájt	billentyűismétlésnél a második, harmadik, . . . várakozás hossza
CLOCK	1 bájt	óra
FLAG	1 bájt	alsó két bitje használt 0. bit ismételt billentyű jelzése 1. bit billentyűre várakozás, illetve érvényes billentyű jelzése
BL	1 bájt	a pufferben lévő adatok száma
FI	2 bájt	a puffer első szabad bájtjára mutat
FO	2 bájt	a puffer első kiolvasandó bájtjára mutat
BE	2 bájt	a puffer végére mutat
BS	2 bájt	a puffer elejére mutat

Ha bármilyen formában átírjuk a megszakításrutinokat, de használni akarjuk a rendszer szolgáltatásait, akkor vigyázni kell arra, hogy amit az eredeti rutin elvégez, azt elvégezze a mi megszakításrutinunk is. Ezt nagyon egyszerűen megtehetjük, ha az EI és RET utasítások helyére egy JP utasítást írunk, amely az eredeti megszakításrutint indítja.

VERHÁS PÉTER

COMMODORE 64

Adalék a rendezéshez

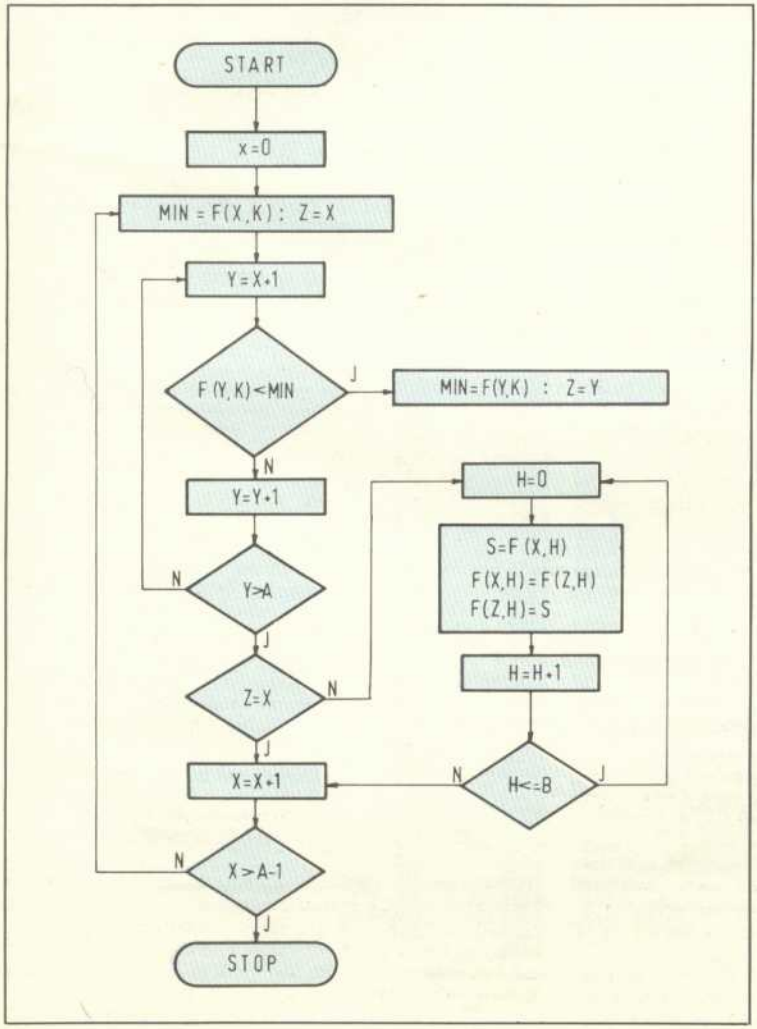
A rendezőprogramok nagy része úgy dolgozik, hogy a szomszédos elemeket kicseréli a feltételnek megfelelően. Amikor már sokalltam az időt a cserélgetésre, elgondolkoztam azon, hogy én ezt hogy csinálnám. Végignéztem tehát a számokon és megkerestem a legkisebbiket, azt kicseréltem az első helyre, és tovább kerestem a maradék számok között. Így a hasonló bonyolultságú programoknál — jelen esetben a buborékmódszernél — négyszer gyorsabban értem el az eredményhez. Remélem, hogy eljárásom közreadásával sok rendezésre fecsérelt időt takarítok meg hasonló gondokkal küszködő társaimnak.

"B" mezőből
"A" számú rekordot tartalmazó
"F" tömb növekvő sorba rendezése
"K" kulcsmező szerint

Program Commodore 64-re

```
1 REM
2 REM   RENDEZÉS
3 REM
10 INPUT"KULCSMEZO";K
20 FOR X=0 TO A-1
30 MIN=F(X,K):Z=K
40 FOR Y=X+1 TO A : IF F(Y,K)
   <MIN THEN MIN=F(Y,K):Z=Y
50 NEXT Y
60 IF ZZ=X THEN 80
70 GOSUB 100
80 NEXT X
90 RETURN
100 FOR H=0 TO B:S=F(X,H):
    F(X,H)=F(Z,H):F(Z,H)=S:NEXT:
    RETURN
```

A rendezés folyamatábrája



Változók

A a rekordok száma
B mezőszám a rekordon belül
K a kulcsmező indexe a rekordban
X futó index, a rendezettség alsó határa
Y a vizsgált rekord indexe
Z a legkisebb kulcsértékű rekord indexe

A program magyarázata

10 Megkérdezi, melyik mező szerint rendezzen
20 Ciklus 0-tól az A—1 rekordig
30 Minimumértéknek felveszi az első rekordot Z=X rekordszámmal
40 A rekordok között megkeresi a legkisebb kulcsértékűt, Z=Y rekordot
60 Ha Z=X, akkor az első rekord kulcsértéke a legkisebb, és nem kell kicserélni
70—100 Ellenkező esetben Z<>X, akkor a Z számú rekordot ki kell cserélni az X számú rekorddal és az X értékét eggyel növelni.

GYŐRI IMRE

SYMPOSIUM ON MOLECULAR ELECTRONICS AND BIOCOMPUTERS

Molekuláris elektronika és bioszámítógép szimpózium Budapest, 1987. augusztus 24—27.

Felvilágosítást ad az Eötvös Loránd Fizikai Társulat 1061 Budapest, Anker köz 1—3. Telefon: 227-040

Bináris keresőalgoritmus C64-re

Különösen adatbázis-kezelési problémák megoldása során gyakran van szükség adatoknak valamely adathalmazból való visszakeresésére. A kínálkozó legegyszerűbb megoldás, hogy az adathalmaz minden elemét, az elsőtől az utolsóig megvizsgáljuk, összehasonlítva azokat a keresett elemmel. Ez azonban különösen BASIC-ben, de más magas szintű programnyelven is meglehetősen lassú. Például egy ezer elemet tartalmazó adatbázis utolsó egyedét néhány perces türelmetlen várakozás után találjuk csak meg.

Természetesen van ennél jobb keresési algoritmus is, nem is egy. A bináris keresési eljárás például még BASIC-ben megírva is az előbbinek töredékére csökkenti a hozzáférési időt. Ilyen algoritmus BASIC programja található például Englisch—Szczepanowski: A VC—1541-es lemezegység programozása (DATA BECKER — NOVOTRADE 1985.) c. könyve 61. oldalán. (Sajnos a könyvnek ez a programja is hibás!) Helyette egy egyszerű kis BASIC programot ismertetek a probléma megoldására *(1. lista)*.

Az M% változóban az adatfájl mérete

```
 10 INPUT"A KERESETT SZO";N$(0)
 20 S=INT(LOG(M%)/LOG(2)+1)
 30 AH%=1
 40 FH%=M%+1
 50 K%=(AH%+FH%)/2
 60 IF N$(0)=N$(K%) THEN PRINT N$(K%):GOTO 10
 70 IF N$(0)<N$(K%) THEN FH%=K%:GOTO 90
 80 AH%=K%
 90 S=S-1:IFS=0 THEN PRINT"NINCS ILYEN":GOTO 10
100 GOTO 50
```

1. lista

2. lista

```
MERET   =$FB
FH      =$FD
AH      =$A9
CIKL    =$FF
AT      =$57
FEL     =$59
SZ01    =$5B
SZ02    =$5D
HOSSZ   =$5F
KITEVO  =$4E
KOZEP   =$4F
CHRGET  =$73
CHROUT  =$FFD2
FRMEVL  =$AD9E
ADRESS  =$B08B
FACADR  =$B7F7
LINOUT  =$AB1E
CIM     =$47
M       =$14
N0      =$26
N1      =$28
NN      =$A7
MAS     =$CFF1
        *=49152
        ;
        JSR CHRGET          ;ELSO VESSZO ATUGRASA
        JSR ADRESS          ;A HIVASBAN MEGJELOLT
        LDA CIM             ;TOMBELEM CIMENEK
        STA N0              ;N0-BA OLVASASA
        LDA CIM+1
        STA N0+1
        LDA N0+1
        STA N1+1
        CLC                 ;EBBOL A KOVETKEZO ELEM
        LDA N0              ;CIMENEK KISZAMITASA
        ADC #3
        STA N1
        BCC 0
        INC N1+1
0       JSR CHRGET          ;2. VESSZO ATUGRASA
        JSR FRMEVL          ;A TOMBMERET BEOLVASASA
        JSR FACADR
        LDA M
        STA MERET
        LDA M+1
        STA MERET+1
        LDA N0              ;AT=N0
        STA AT
        LDA N0+1
        STA AT+1
        LDY #1
        LDA (AT),Y          ;SZ01-BEN A KERESETT
        STA SZ01            ;SZO LEIROBOL KIEMELT
        INY                 ;ABSZOLUT CIME
        LDA (AT),Y
        STA SZ01+1
        LDA #15             ;A STRINGEK
        STA HOSSZ           ;HOSSZA -> HOSSZ-BA
        LDA #1
        STA KOZEP           ;KOZEP=1
        LDA #0
        STA KITEVO          ;KITEVO=0
        STA KOZEP+1
HATV    INC KITEVO          ;KITEVO=KITEVO+1
        ASL KOZEP
        ROL KOZEP+1         ;KOZEP = KOZEP * 2
        SEC
        LDA KOZEP           ;IF KOZEP < MERET
        SBC MERET           ;THEN MEG EGY CIKLUS
        LDA KOZEP+1
        SBC MERET+1
        BCC HATV            ;UGRAS HATV-RA
        LDA MERET           ;FH=MERET
        STA FH
        LDA MERET+1
        STA FH+1
        LDA #0
        STA AH              ;AH=0
        STA AH+1
        STA CIKL            ;CIKL=0
HUROK   INC CIKL            ;CIKL=CIKL+1
        LDA KITEVO
        CMP CIKL
        BMI HID
        CLC                 ;KOZEP=FH+AH
        LDA FH
        ADC AH
        STA KOZEP
        LDA FH+1
        ADC AH+1
        STA KOZEP+1
        LSR KOZEP+1         ;KOZEP=KOZEP/2
        ROR KOZEP           ;KOZEP=FELEZO SORSZAM
        LDA KOZEP+1         ;NN=KOZEP - 1
        STA NN+1
        SEC
        LDA KOZEP
        SBC #1
        STA NN
        BCS FELEZ
        DEC NN+1
        LDA N1              ;AT=N1
        STA AT
        LDA N1+1
        STA AT+1
FELEZ   LDA NN              ;FEL=NN-HEZ TARTOZO CIM
        STA FEL
        LDA NN+1
        STA FEL+1
        ASL FEL
        ROL FEL+1
        ASL FEL
        ROL FEL+1
        SEC
        LDA FEL
        SBC NN              ;FEL=3*NN+AT
        STA FEL             ;A KOZEPSO ELEM
        LDA FEL+1           ;LEIROJA CIMENEK
        SBC NN+1            ;KISZAMITASA
        STA FEL+1
        CLC
        LDA FEL
        ADC AT
        STA FEL
        LDA FEL+1
        ADC AT+1
        STA FEL+1
        LDY #1              ;A MINDENKORI KOZEPSO SZ
        LDA (FEL),Y         ;ABSZOLUT CIME SZ02-BE
        STA SZ02
        INY
        LDA (FEL),Y
        STA SZ02+1
        LDY #0              ;OSSZEHASONLITAS
HSNL    LDA (SZ01),Y
        CMP (SZ02),Y
        BCC KISEBB          ;HA A KERESETT KISEBB
        BEQ TOVABB          ;HA EGYENLO
        JMP NAGYBB          ;HA NAGYOBB
HID     BMI SEHOL
TOVABB  INY
        CPY HOSSZ           ;IF Y=HOSSZ
        BEQ KIIR            ;THEN KIIRATAS
        JMP HSNL            ;KULONBEN VISSZAUGRAS
KISEBB  LDA KOZEP           ;FH=KOZEP
        STA FH
        LDA KOZEP+1
        STA FH+1
        JMP HUROK
NAGYBB  LDA KOZEP           ;AH=KOZEP
        STA AH
        LDA KOZEP+1
        STA AH+1
        JMP HUROK
SEHOL   LDX #$00            ;A "NINCS ILYEN"
T       LDA TXT,X           ;UZENET KIIRATASA
        JSR CHROUT
        INX
        CPX #$0B
        BNE T
        RTS
TXT     .TEXT "NINCS ILYEN"
KIIR    LDY #0              ;A MEGTALALT SZO
KIR1    LDA (SZ02),Y        ;MAS-BA MASOLASA
        STA MAS,Y
        INY
        CPY HOSSZ
        BNE KIR1
        LDA #MASC
        LDY #MASC
        JSR LINOUT          ;KIIRATASA
        RTS
        .END

ZEILEN:173   SYMBOLE:37   FEHLER:0
```

van. Az N$ tömbbe kerül beolvasásra az adathalmaz.

A bináris keresés alkalmazhatóságának előfeltétele, hogy rendezett adathalmazunk legyen. Ez azt jelenti, hogy például nagyság szerint, vagy éppen ábécébe kell rendeznünk az adathalmaz elemeit. A rendezés hatékony megvalósítása viszont legalább akkora probléma, mint a keresés. Erre vi-szont Erdős Iván: COMMODORE 64 AS-SEMBLY c. könyvében találunk megbízható algoritmusokat.

A program gondolatmenete a következő.

1. Felezzük meg rendezett adathalmazunkat.

2. A keresett elemet hasonlítsuk össze a felező elemmel.

3. Ha egyezik, akkor a keresésnek vége.

4. Ha kisebb, akkor az adathalmaz alsó felébe, különben a felsőbe esik a keresett elem.

5. Ismételjük meg az eljárást az 1. lépéstől úgy, hogy a felezés mindig a maradék halmazra vonatkozzon.

Ezzel az eljárással egy M elemű tömb bármely eleme legfeljebb S=INT(LOG² M+1), azaz S=INT(LOG(M)/LOG² +1) lépésben elérhető. Így például az 1000. elem ezer keresőlépés helyett tizenegy lépésben elérhető. (S a lépésszám.)

A következőkben egy assembly nyelvű bináris keresőalgoritmust szeretnék bemutatni, amely nagy adatbázisok hatékony kezelésében nyújthat segítséget. A program *(2. lista)* logikai megfelelője a közölt BASIC programnak, s így egy karakteres tömb valamely elemének megkeresésére szolgál. Hívása BASIC-ből: SYS 49152,N$(Ø),M+1, ahol N$(Ø) a karakteres tömbnek az az eleme, amelybe a keresett szót kell elhelyezni. M+1 a tömb elemeinek száma, ha a Ø. elemet is számításba vesszük.

Az egyszerűség kedvéért a tömbelemek első 15 karakterét használjuk fel azonosításra. Ez azonban tetszőlegesen választható. Természetesen a tömböt előzetesen e szerint kell feltölteni. A tömbelemek egységes hosszát BASIC-ből az N$(I)=LEFT$(N$(I)+" ",15) értékadással érhetjük el. A kereső szót is ilyen alakra kell hozni:

```
INPUT K$
N$(Ø)=LEFT$(K$+"                ",15)
```

A program működése

35Ø—51Ø	A SYS paraméterek átvétele: a Ø. és az 1. tömbelem címének kiszámítása az ADRESS interpreter rutin segítségével, és a tömbméret beolvasása.
52Ø—59Ø	Munkaváltozók beállítása.
6ØØ—65Ø	N$(Ø) leírójából a keresett szó abszolút címét adó 2. és 3. bájtot SZO1-be töltjük. (A leírók hárombájtosak, az első, Ø. bájt a sztring hosszát, a második és harmadik, azaz az 1. és 2. sorszámú bájt a sztring szövegének a címét mutatja.)
66Ø	Itt kell beállítani a szóhosszt.
69Ø—81Ø	Megkeressük 2-nek M-nél kisebb, legnagyobb hatványát.
82Ø—122Ø	A tömb felezése, részhalmazokra bontása.
123Ø—136Ø	A felező elem leírója címének kiszámítása.
137Ø—142Ø	A felező elem abszolút címének kiemelése a leíró 1. és 2. bájtjából és SZO2-be tétele.
143Ø—153Ø	A keresett szó: N$(Ø), és a mindenkori középső szó összehasonlítása.
154Ø—163Ø	Alprogramok, melyek a részhalmazok határait állítják be.
164Ø—171Ø	A "NINCS ILYEN" üzenet kiírása a CHROUT KERNAL rutin segítségével.
172Ø—181Ø	A megtalált szót $CFF1-től $CFFF-ig terjedő területre másoljuk, és a LINOUT interpreter rutin segítségével a képernyőre íratjuk. (Erre más lehetőség is van.)

A 93Ø—149Ø—164Ø sorok ugrási hidat képeznek. A 93Ø-as sorban kijelölt ugrási cím (SEHOL címkével az 164Ø-es sorban) 128 bájtnál távolabb van. Ezért a közvetlen áthidalás nem lehetséges.

A megtalált szó képernyőre íratása itt természetesen egyszerűsítés a program kedvéért. Gyakorlatban általában nem ez a feladat.

Tételezzünk fel például egy 13ØØ—15ØØ rekordot tartalmazó relatív fájlt. Tartalmazzák a rekordok például dolgozók személyi adatait. Legyen az első mező a dolgozó neve. Tekintsük a név azonosításához elegendő első 15 karaktert keresési kulcsnak.

A relatív fájl egészének rendezettségét biztosítani körülményes, de a kulcsokból egy álladóan rendezett indexállományt hozhatunk létre egy tömbváltozóban. Ez a nevek mellett a rekord sorszámát tartalmazza.

A keresés erre a rendezett halmazra vonatkozik. Ha megtaláltuk a keresett nevet, kiolvassuk belőle a relatív rekord sorszámát, és az ADRESS rutin segítségével például egy R$ nevű változóba másoljuk. Az R$-ből BASIC-ben átvesszük a sorszámot, s a relatív fájlkezelésben ismert eljárás szerint meghívjuk a kívánt rekordot.

Ezzel kapcsolatban azonban ki kell elégítenünk két követelményt:

— R$-et a BASIC programban előzetesen definiálni kell. Például: R$="". Majd egy látszólagos műveletet kell vele végezni, így: R$=R$+" (legalább 20 szóköz) ".

— A rutin meghívásakor a SYS-ben nem elegendő csak az indexállomány nevét és méretét megadni, hanem R$-re is hivatkozni kell.

Végezetül néhány mérési adat a futásidők összehasonlítására. Az időmérést egy 1300 elemet tartalmazó tömbben való keresésnél végeztem.

Soros keresésnél az elérési idő a keresett elem sorszámával nagyjából arányosan nő. Az erzediken túli elemek elérése memóriában tárolt adatoknál 1 és 2 perc között van. Relatív fájlban való soros keresés esetén az ezrediken túli elemek megtalálása 6 percnél is hosszabb időt vesz igénybe.

A bemutatott BASIC programmal történő kerestetés a memóriában levő tömbben 18 és 25 másodperc közötti elérési időt ad.

Az assembly rutin használata esetén a meghívás és a sikeres visszatérés közötti idő 0,025 és 0,033 másodperc közé esik.

Ha az assembly rutint egy relatív fájlhoz kapcsolódó indexfájlban való kerestetésre használom, a meghívás és a relatív fájl rekord formázott képernyőre írása közötti idő 6—8 másodperc.

DR. JÁNOSA ANDRÁS

Papírszélesítés

Seikosha GP—50 S és ZX-printer tulajdonos spectrumosok használhatják jól ezt a kis programot.

Az említett printerek elég keskeny papírra dolgoznak, általában csak 32 betűt tudnak egy sorba kiírni, 256 pixel szélességben. A Tasword típusú szövegszerkesztők a másik végletbe esnek: épp hogy felismerhető, de már nehezen olvasható sovány betűket használnak, így soronként 64 karaktert értek el.

Megpróbáltam átmenetet találni a 32 és a 64 között. A legcélszerűbbnek és a legérdekesebbnek a változó karakterszélesség látszott — gondoljunk csak az m és az i közötti különbségre. Így ugyan nehezebb szépen formázott szövegeket készíteni, de minden betű jól olvasható, és soronként átlagosan 50—60 karakter fér el.

A program a szöveget a grafikus képernyőre írja ki, az egyszerűség érdekében PLOT-tal, de még így is meglepően gyors. Ha a képernyő megtelik, COPY-val kimásolja a papírra, majd törli a képernyőt, és így tovább. Torzítás nélkül nyomtatja a lelógó írásjeleket és betűket is. Ennek módszerét a Primótól kölcsönöztem.

A programot eredetileg a SPECTRE-MAC nevű makrofeldolgozó fájljainak kiírásához készítettem, ezért csupán annak karaktereit ismeri fel, a b7 bitet nem veszi figyelembe:

ØCH : kurzor home
ØDH : kurzor az aktuális sor elejére (Old-Line-CR)
1CH : 32 pixelenkénti tabulátor
1FH : kurzor le (soremelés)
2ØH—7FH : a Spectrum ASCII karakterei

A többi ØØH és 1EH közötti karakterrel semmit nem csinál, még "?"-t sem nyomtat. A többi paramétert az *1. táblázat* tartalmazza.

A program természetesen módosítható, például újabb vezérlőkarakterek felismerésére, a betűkészlet bővítésére. A már említett változó betűszélesség miatt a karakterkészlet azonban nem túl könnyen variálható. Az adatszerkezetet a *2. táblázat* tartalmazza. Aki komolyabban akar bővítéssel foglalkozni, eljátszhat a disassemblerrel is, hisz maga a program alig száz sor, de — korlátozott számban — levélben szívesen elküldöm a forráslistát.

A szubrutin BASIC-ből is kényelmesen használható a Spectrum megfelelő csatornacímének átírásával:

```
LET a=PEEK 23631+PEEK 23632*256+3*5
POKE a,188: REM kezdőcím alsó bájt
POKE a+1,252: REM felső bájt
```

Ezután PRINT #3-mal és LPRINT-tel a keskenyített betűkkel írhatunk a nyomtatóra. A printer eredeti üzemmódját POKE a,244: POKE a+1,9 segítségével állíthatjuk vissza. Az *a* változó értéke általában 23749, de csak ha nincs Interface—1 a géphez kapcsolva.

A program beírása a következőképpen történik. Az *1. listán* látható segédprogrammal begépeljük a *2. listán* látható hexadecimális számokat (az ellenőrző összegekkel együtt), az esetleges hibákat kijavítjuk,

majd a programot *kipróbálás előtt* magnóra vesszük:
SAVE "MacPrint"CODE 64700,668

A hexa listák beírásának bevált módszere, hogy saját pötyögési sebességünknek megfelelően magnóra mondjuk a számokat, 4 bájtonként kisebb, 8 bájtonként nagyobb szüneteket tartva. Ezt füllel ellenőrizzük, majd magnóról begépeljük. A sorok elején célszerű a címet is mondani.

Még egy megjegyzés: a program a nyomtatási pozíciót a COORDS rendszerváltozóban tárolja, így a képernyőn bárhová pozicionálhatunk egy
PLOT INVERSE 1;x,y
utasítással.

1. táblázat. A szubrutin jellemzői

A kiírandó karakter kódja: A regiszter
Megváltoztatja: B' regiszter
64700 : kezdőcím és belépési pont
64869 : szabad terület bővítésekre
64908 : karakterkészlet kezdete
65367 : karakterkészlet utolsó bájtja

Módosítható jellemzők:
64761 : sortávolság, alap: 12
64768 : lapmagasság, alap: 168. Szép nyomtatási képet akkor kapunk, ha a lapmagasság a sortávolság többszöröse, és legközelebb van a maximális 176-hoz.

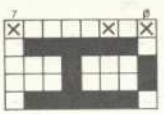
A programban levő abszolút címzések helye:

64727-en: 64785 (CALL KERES)
64730-on: 64805 (CALL KIIR)
64786-on: 64908 (LD HL, CHARS)
64862-n: 64700 (CALL PROUT)
64867-en: 64700 (JP PROUT)

Ezek módosításával a programot "gyalog" (BASIC programmal) relokálhatjuk. 16 k-s spectrumosok, figyelem!

2. táblázat. Az adatszerkezet

A karakterkészlet 96 *tételből* áll
Egy tétel részei: *fej, törzs*
A fej : 1 bájt; b7 = 1
b6 = 1, ha lelógó írásjel-jegy (,;gjpgy)
b5—b0 = *n* : a tétel hossza bájtokban
A törzs : n—1 bájt; a karaktermátrix



Példa:
Az "A" betű tétele:
85H,7EH,11H,11H,7EH
A "g" betű tétele:
C5H,0EH,51H,51H,3FH

MÉSZÁROS LÁSZLÓ

```
    1 DEF FN a$(i)=CHR$ (i+48+7*(
i>9))
    2 DEF FN b$(i)=FN a$(INT (i/1
6))+FN a$(i-16*INT (i/16))
    3 DEF FN c$(i)=FN b$(INT (i/2
56))+FN b$(i-256*INT (i/256))
    4 DEF FN a(i$)=CODE i$-48-7*(
i$(1)>"9")
    5 DEF FN b(i$)=FN a(i$(1))*16
+FN a(i$(2))
    6 DEF FN c(i$)=FN b(i$(1 TO 2
))*256+FN b(i$(3 TO 4))
   10 CLEAR 64599
  100 DIM a$(27)
  110 FOR i=64700 TO 65367 STEP 8
  120 INPUT (FN c$(i);":")a$
  130 LET s=0
  140 FOR j=0 TO 7
  150 LET c=FN b(a$(3*j+1 TO 3*j+
2))
  160 LET s=s+c
  170 POKE i+j,c
  180 NEXT j
  190 LET st=FN c("0"+a$(25 TO ))
  200 IF s<>st THEN BEEP 2,2: PRI
NT a$: GO TO 120
  210 PRINT FN c$(i);" ok."
  220 NEXT i
  230 PRINT "[redacted]"
  240 SAVE "MacPrint"CODE 64700,6
68
  250 VERIFY ""CODE
```

1. lista

2. lista

```
FC8C: F5 D5 E5 C5 E8 7F FE 0C,5E3
FC94: 28 42 FE 00 28 17 FE 1F,2C1
FC9C: 28 23 FE 1C 28 15 FE 20,2C1
FCA4: 38 08 CD 11 FD CD 25 FD,408
FCAC: C1 E1 D1 F1 C9 3E 00 32,49D
FCB4: 7D 5C 18 F4 3A 7D 5C C6,38E
FCBC: 20 E6 E0 18 F2 3A 7E 5C,404
FCC4: FE 16 38 07 D6 0C 32 7E,2E5
FCCC: 5C 18 DD 06 A8 F3 CD AF,46E
FCD4: 0E CD 5B 0D 01 00 AF ED,2F0
FCDC: 43 7D 5C 18 CB 21 8C FD,3A9
FCE4: 16 00 D6 20 28 08 47 7E,201
FCEC: E6 0F 5F 19 10 F9 7E 23,317
FCF4: C9 ED 4B 7D 5C 50 CB 77,46C
FCFC: 28 02 05 05 58 E6 0F D9,25A
FD04: 3D 47 D9 7E CB 3F 30 08,320
FD0C: C5 D5 E5 F5 CD E8 22 F1,63C
FD14: E1 D1 C1 28 03 05 18 EC,3A7
FD1C: 0C 43 23 D9 10 E4 D9 0C,324
FD24: 79 32 7D 5C FE FA D8 3E,492
FD2C: 0D CD 8C FC 3E 1F C3 8C,46E
FD34: FC 00 00 00 00 00 00 00,0FC
FD3C: 00 00 00 00 00 00 00 00,000
FD44: 00 00 00 00 00 00 00 00,000
FD4C: 00 00 00 00 00 00 00 00,000
FD54: 00 00 00 00 00 00 00 00,000
FD5C: 00 00 00 00 00 00 00 00,000
FD64: 00 00 00 00 00 00 00 00,000
FD6C: 00 00 00 00 00 00 00 00,000
FD74: 00 00 00 00 00 00 00 00,000
FD7C: 00 00 00 00 00 00 00 00,000
FD84: 00 00 00 00 00 00 00 00,000
FD8C: 84 00 00 00 84 00 5F 00,167
FD94: 84 03 00 03 86 24 7E 24,1D5
FD9C: 7E 24 86 2E 2A 7F 2A 3A,263
FDA4: 86 63 13 08 84 63 86 32,283
FDAC: 4D 4A 30 50 83 02 01 84,221
FDB4: 1C 22 41 84 41 22 1C 85,207
FDBC: 2A 1C 1C 2A 84 08 3E 08,15E
FDC4: C3 20 18 84 08 08 08 82,219
FDCC: 40 84 60 1C 03 85 3E 51,257
FDD4: 49 3E 84 42 7F 40 85 62,2F3
FDDC: 51 49 46 85 23 41 45 38,249
FDE4: 85 38 26 71 20 85 27 45,255
FDEC: 45 39 85 3E 45 45 38 85,288
FDF4: 03 81 19 07 85 36 49 49,1D1
FDFC: 36 85 0E 51 51 3E 83 00,22C
FE04: 24 C3 20 19 85 08 14 22,1E3
FE0C: 41 85 24 24 24 24 85 41,21C
FE14: 22 14 08 85 02 51 09 06,125
FE1C: 85 3E 41 59 1E 85 7E 11,28F
FE24: 11 7E 86 41 7F 49 49 36,29D
FE2C: 85 3E 41 41 22 86 41 7F,2AD
FE34: 41 41 3E 85 7F 49 49 41,297
FE3C: 85 7F 09 09 01 85 3E 41,218
FE44: 51 72 85 7F 08 08 7F 84,2DA
FE4C: 41 7F 41 85 20 40 40 3F,265
FE54: 85 7F 08 14 83 85 7F 40,2C7
FE5C: 40 40 86 7F 04 08 04 7F,214
FE64: 85 7F 04 08 7F 85 3E 41,293
FE6C: 41 3E 85 7F 11 11 0E 86,238
FE74: 3E 41 51 7E 85 7F 11 31,2A4
FE7C: 4E 85 26 49 49 32 86 01,244
FE84: 01 7F 01 01 85 3F 40 40,1C6
FE8C: 3F 85 0F 30 40 30 0F 85,209
FE94: 3F 40 30 40 3F 85 53 14,228
FE9C: 08 14 53 85 07 08 78 08,194
FEA4: 07 86 61 51 49 45 43 84,294
FEAC: 7F 41 41 84 03 1C 60 84,288
FEB4: 41 41 7F 84 06 7F 06 84,294
FEBC: 40 40 40 85 64 7E 45 41,2AD
FEC4: 85 20 54 54 78 85 7F 44,30D
FECC: 44 38 84 38 44 44 85 38,27D
FED4: 44 44 7F 85 38 54 54 18,284
FEDC: 84 04 7E 05 C5 0E 51 51,280
FEE4: 3F 85 7F 04 04 78 84 44,288
FEEC: 7D 40 C3 41 3F 85 7F 10,314
FEF4: 28 44 83 3F 40 86 7C 04,274
FEFC: 7C 04 78 85 7C 04 04 78,279
FF04: 85 38 44 44 38 C5 7F 11,2D2
FF0C: 11 0E C5 0E 11 11 7F 86,218
FF14: 7C 04 04 08 85 48 54 54,201
FF1C: 20 84 04 3F 44 85 3C 40,22C
FF24: 40 7C 86 1C 20 40 20 1C,1FA
FF2C: 86 3C 40 30 40 3C 86 44,278
FF34: 28 10 28 44 C5 0F 50 50,218
FF3C: 3F 85 54 54 4C 44 84 08,298
FF44: 36 41 82 7F 84 41 36 08,278
FF4C: 85 04 02 04 02 87 3E 41,197
FF54: 5D 56 41 3E 00 3C 42 42,1F1
```

A VIC—20 tárcímei

A VIC—20-as géppel rendelkező olvasóink részére közreadjuk a VIC REVEALED, NICK HAMPSIRE, 1981. c. könyv alapján a legfontosabb tárcímeket, valamint a BASIC és a KERNAL ROM rutinokat.

POKE címek	Magyarázat
0—2	USR funkció helye
3—4	lebegőpontosból egész számmá konvertálás
5—6	egészből lebegőpontos számmá konvertálás
13	változó jelző: 0 = numerikus, 255 = sztringváltozó
14	egész jelző: 0 = lebegőpontos, 128 = egész szám
69—70	aktuális változónév
71—72	aktuális változó kezdőcíme
147	load vagy verify jelző
152	nyitva levő fájlok száma
157	RUN (=0), vagy direkt (=80) módban dolgozik a gép
160—162	óra (TI$)
183	az aktuális fájlnév hossza
198	a billentyűpufferben levő karakterek száma
199	inverzmód jelző: 0 = ki, 18 = be
201—202	kurzorhely (sor, oszlop sorrendben)
203	az éppen lenyomott billentyű kódja (=64, ha nincs billentyű)
204	kurzorvillogás-engedélyező jelző (0 = be, 1 = ki)
215	az utoljára lenyomott billentyű ASCII kódja
251—255	szabad zérólapos bájtok
256—511	processzor veremterülete
512—600	BASIC input puffer
601—610	logikaifájlszám-táblázat
611—620	egységszámtáblázat
621—630	másodlagos cím vagy R/W táblázat
631—640	billentyűzetpuffer
650	billentyűismétlés jelző: 0 = csak a kurzorirányítók 128 = mindegyik billentyű ismétel
651	késleltetés, mielőtt az ismétlés megtörténik
652	ismétlések közötti szünet
653	shift, CTRL, valamint C= jelzőbájt
657	shift+C= váltást engedélyező(0 = engedélyezve, 128 = tiltva)
673—767	szabad terület

INDIREKT UGRÁSI CÍMEK

768—769	hibakiíró rutin (x regiszterbe a hiba kódja)
770—771	BASIC melegindítás
772—773	tokennné alakítás
774—775	a memóriában levő soroknak lista formájú alakítása
776—777	következő utasítás végrehajtása
778—779	numerikus változó bevitele FAC-ba
780	akku átadása a SYS utasítás alatt
781	x regiszter átadása a SYS utasítás alatt
782	y regiszter átadása a SYS utasítás alatt
783	állapotregiszter átadása a SYS utasítás alatt

KERNAL MUTATÓK CÍMEI

788—789	IRQ RAM vektora
790—791	BRK RAM vektora
792—793	NMI RAM vektora (RESTORE billentyű hatására)
794—795	OPEN logikai fájl vektora
796—797	CLOSE logikai fájl vektora
798—799	CHKIN KERNAL rutin vektora
800—801	CHKOUT KERNAL rutin vektora
802—803	CLRCHN KERNAL rutin vektora
804—805	CHRIN KERNAL rutin vektora
806—807	CHROUT KERNAL rutin vektora
808—809	STOP KERNAL rutin vektora
810—811	GETIN KERNAL rutin vektora
812—813	CLALL KERNAL rutin vektora
814—815	BASIC USR vektora
816—817	LOAD vektora
818—810	SAVE vektora
820—827	szabad terület
828—1020	kazettapuffer
1024—4095	3 k RAM bővítés helye
4096—7679	felhasználói BASIC terület
7680—8191	képernyőmemória
8192—16383	8 k RAM/ROM bővítés 1
16384—24575	8 k RAM/ROM bővítés 2
24576—32767	8 k RAM/ROM bővítés 3

MEGJEGYZÉS: Ha valamely bővítéssel dolgozik a gép, akkor a felhasználói RAM-terület elhelyezkedése a következőképpen módosul:

4096—4607	képernyőmemória
4608—??	felhasználói BASIC terület
37888—38399	színmemória

32768—36863	karaktergenerátor (ROM)
36864—37877	I/O BLOCK 1
36864	0—6 bitek állítják a képernyő vízszintes helyzetét
36865	a képernyő függőleges helyzetét állítja
36866	0—6 bitek a képernyő oszlopainak számát tartalmazzák
36867	0—6 bitek a képernyő sorainak számát tartalmazzák
36868	TV RASTER sorát adja
36869	0—3 bitek a karaktermemória helyzetét adják 4—7 bitek a video eredeti kezdőcímét adják
36870	a fényceruza vízszintes pozícióját adja
36871	a fényceruza függőleges pozícióját adja
36874—36878	zenei bájtok
36879	0—2 bitek a keret színét állítják a 3. bit választ inverz vagy normál mód között 4—7 bitek a háttér színét állítják
37136—37151	6522 chip PIA 1.
37152—37167	6522 chip PIA 2.
37888—38399	színmemória, ha van valamely bővítés
38400—38911	színmemória, ha bővítés nélküli a gép
38912—39935	I/O BLOCK 2
39936—40959	I/O BLOCK 3
40960—49151	8 k ROM bővítés
49152—57343	8 k BASIC ROM
57344—65535	8 k KERNAL ROM

A BASIC INTERPRETER FONTOSABB RUTINJAI

Cím (hexadecimálisan)	*Magyarázat*
C000—C045	az elsődleges kulcsszavak ugrási címei
C046—C073	egyes funkciók ugrási címei
C074—C091	az operációs rendszer szerveződési és ugrási címei
C092—C192	BASIC kulcsszavak táblázata
C193—C2A9	BASIC üzenetek, legtöbbjük hibaüzenet
C38A—C3B7	FOR vagy GOSUB részére helyet készít a veremben
C3B8—C3FA	űr felnyitása a memóriában
C3FB—C407	teszt: a verem nem túl mély-e
C408—C434	a rendelkezésre álló memória méretének ellenőrzése
C435—	a megadott hibaüzenet kiírása, majd:
C474—C482	PRINT READY
C483—C532	új BASIC programsor a billentyűzetről
C533—C55F	programsorok átépítése a memóriában
C560—C57B	egy sor elfogadása a billentyűzetről
C57C—C612	kulcsszavak tokenjükre cserélése
C613—C641	megkeresi az adott sorszámnak megfelelő sort
C642—	NEW végrehajtása, majd:
C660—C68D	CIR végrehajtása
C68E—C69B	BASIC mutatók visszaállítása programkezdetre
C69C—C741	LIST végrehajtása
C742—C7EC	FOR végrehajtása
C7ED—C819	BASIC kijelentés végrehajtása
C81D—C82B	RESTORE végrehajtása
C82C—C856	STOP és END végrehajtása
C857—C870	CONT végrehajtása
C871—C882	RUN végrehajtása
C883—C89F	GOSUB végrehajtása
C8A0—C8D1	GOTO végrehajtása
C8D2—C8EA	RETURN végrehajtása
C8EB—C905	DATA végrehajtása, azaz a következő utasításra ugrás
C906—C908	következő BASIC utasítás keresése
C909—C927	következő BASIC sor keresése
C928—C93A	IF végrehajtása
C93B—C94A	REM végrehajtása, azaz a következő BASIC sorra ugrás
C94B—C96A	ON végrehajtása
C96B—C9A4	fixpontos szám átvétele BASIC-ből
C9A5—CA1C	LET végrehajtása

CA1D—CA2B egy ASCII szám hozzáadása akkumulátor 1-hez
CA2C—CA7F LET végrehajtásának folytatása
CA80—CA85 PRINT# végrehajtása
CA86—CA99 CMD végrehajtása
CA9A—CB1D PRINT végrehajtása
CB1E—CB3A a memóriából egy sztring kiírása
CB3B—CB4C HOME, CURSOR FEL, CURSOR LE, ... stb. karakterek kiírása
CB4D—CB7A rossz input adatok kezelése
CB7B—CBA4 GET végrehajtása
CBA5—CBBE INPUT# végrehajtása
CBBF—CBF8 INPUT végrehajtása
CBF9—CC05 input elfogadása
CC06—CCFB READ végrehajtása, egyes rutinjait a GET és INPUT is használja
CCFD—CC1D üzenetek: EXTRA IGNORED, REDO FROM START
CD1E—CD77 NEXT végrehajtása
CD78—CD9D adattípus ellenőrzése, REDO FROM START kiírása, ha kell
CD9E—CEF0 valamilyen numerikus vagy sztring kifejezés kiértékelése
CEF1—CEF6 zárójelen belüli kifejezés kiértékelése
CEF7—CEF9 a jobb (csukó) zárójel ellenőrzése
CEFA—CEFC a bal (nyitó) zárójel ellenőrzése
CEFD—CF07 a vessző ellenőrzése
CF08—CF0C kiírás: SYNTAX ERROR és kilépés
CF0D—CF13 a leendő kiértékelésekhez szükséges funkciók felállítása
CF14—CFA6 változónév keresése
CFA7—CFE5 azonosít és utalások felállítása
CFE6—CFE8 OR végrehajtása
CFE9—D015 AND végrehajtása
D016—D07D összehasonlítás: sztring vagy numerikus
D07E—D08A DIM végrehajtása
D08B—D112 változó elhelyezkedésének keresése
D113—D11C vizsgálat: betű-e az ASCII karakter
D11D—D193 új BASIC változó elhelyezkedése a memóriában
D194—D1A4 a mutatókat elrendező szubrutin
D1A5—D1A9 32768 lebegőpontos alakban
D1AA—D1D0 egy pozitív egész kifejezés kiértékelése
D1D1—D34B egy elrendezés létrehozása vagy keresése
D34C—D37C index alapján a tömb méretét számolja ki
D37D—D390 FRE végrehajtása, majd:
D391—D39D konvertálás fixpontosból lebegőpontossá
D39E—D3A5 POS végrehajtása
D3A6—D3B2 ellenőrzi, hogy direkt módban van-e a gép, print ILLEGAL DIRECT
D3B3—D3E0 DEF végrehajtása
D3E1—D3F3 FN x szintaxisának ellenőrzése
D3F4—D464 FN x kiértékelése
D465—D474 STR$ végrehajtása
D475—D486 sztring vektor kiszámítása
D487—D4F3 ellenőrzi és felállítja a sztringet
D4F4—D525 a sztringmutatókat felépítő szubrutin
D526—D5BC hulladékgyűjtő szubrutin (sztringeknél)
D5BD—D605 ellenőrzése annak, hogy jó-e a memóriában levő sztring
D606—D63C sztring összeállítása
D63D—D679 sztringek összeláncolása
D67A—D6A2 sztring felépítése a memóriában
D6A3—D6DA a nemkívánatos sztring törlése
D6DB—D6EB a verem törlése
D6EC—D6FF CHR$ végrehajtása
D700—D72B LEFT$ végrehajtása
D72C—D736 RIGHT$ végrehajtása
D737—D760 MID$ végrehajtása
D761—D77B sztringparamétereket kiveszi a veremből
D77C—D781 LEN végrehajtása
D782—D78A sztringmódról numerikus módra áll át
D78B—D79A ASC végrehajtása
D79B—D7AC input bájt paraméter
D7AD—D7EA VAL végrehajtása
D7EB—D7F6 2 paraméter elrakása POKE vagy WAIT számára
D7F7—D80C lebegőpontosból fixpontossá konvertálás
D80D—D823 PEEK végrehajtása
D824—D82C POKE végrehajtása
D82D—D848 WAIT végrehajtása
D849—D84F 0.5 hozzáadása akkumulátor 1-hez
D850—D861 kivonás végrehajtása
D862—D946 összeadás végrehajtása
D947—D97D akkumulátor 1. kiegészítése
D97E—D982 kiírás: OVERFLOW, és kilépés
D983—D9BB egy bájt sokszorosítása
D9BC—D9E9 állandók: 1, SQR (0.5), SQR (2) stb.
D9EA—DA2F LOG végrehajtása
DA30—DA58 szorzás végrehajtása
DA59—DA8B bitenkénti szorzás szubrutinja
DA8C—DAB6 akkumulátor 2-be tölt a memóriából
DAB7—DAD3 akkumulátor 1. és 2. tesztelése
DAD4—DAE1 túlcsordulás és alulcsordulás kezelése
DAE2—DAF8 szorzás 10-zel
DAF9—DAFD 10 lebegőpontos alakban
DAFE—DB06 osztás 10-zel
DB07—DB11 osztás végrehajtása
DB12—DBA1 osztás valamely számmal
DBA2—DBC6 akkumulátor 1-be tölt a memóriából
DBC7—DBFB akkumulátor 1-ből a memóriába tölt
DBFC—DC/0B akku 2-t akku 1-be másol
DC0C—DC1A akku 1-et akku 2-be másol
DC1B—DC2A akku 1. kerekítése
DC2B—DC38 kiszámolja akku 1. előjelét
DC39—DC57 SGN végrehajtása
DC58—DC5A ABS végrehajtása
DC5B—DC9A összehasonlítja akku 1-et a memória tartalmával
DC9B—DCCB lebegőpontosból fixpontosá konvertálás
DCCC—DCF2 INT végrehajtása
DCF3—DD7D sztringből lebegőpontossá konvertálás
DD7E—DDB2 új ASCII számjegy elrakása
DDB3—DDC1 sztringkonverziós állandó: 99999999,999999999
DDC2—DDCD kiírás: IN; követi:
DDCD—DDDC BASIC sorszám
DDDD—DF10 TI$ számmá konvertálása
DF11—DF70 állandók a numerikus konverziók részére
DF71—DF77 SQR végrehajtása
DF78—DFB3 energiafunkció ellátása
DFB4—DFBE NOT végrehajtása
DFBF—DFEC állandók, sztringértékeléshez
DFED—E03F EXP vegrehajtása
E040—E089 funkciósorozat-kiértékelő szubrutin
E08A—E093 manipulációs állandó RND részére
E094—E0F5 RND végrehajtása
E0F6—E260 KERNAL ROM toldás a BASIC ROM-ban
E261—E267 COS végrehajtása
E268—E2B0 SIN végrehajtása
E2B1—E2DC TAN végrehajtása
E2DD—E30A állandók a trigonometriai számításokhoz (pi, 2*pi, 0.25, ...)
E30B—E33A ATN végrehajtása
E33B—E377 állandók ATN számításokhoz
E378—E386 RAM vektorok inicializálása
E387—E3A3 CHARGET rutin másolata
E429—E44E üzenet: BYTES FREE, ****CBM BASIC V2 ****
E44F—E47B vektorok inicializálása
E47C—E4FF üres terület

KERNAL RUTINOK

E500—E504 A 6522 chip címével tér vissza
E505—E509 a képernyő oszlop- és sorszámával tér vissza
E50A—E517 olvassa/állítja a kurzor pozícióját
E518—E580 I/O eszközök inicializálása
E581—E586 HOME funkció
E587—E5B4 a kurzor elmozdítása az aktuális sorban
E5B5—E5C2 NMI rutin kezdete (RESTORE billentyű megnyomása esetén)
E5C3—E5CE 6560 CHIP IN inicializálása
E5CF—E6C4 egy karakter elmozdítása a sorban
E64F—E64E egy sor bevitele, „RETURN"-ig
E742—E8E7 PRINT rutin
E8E8—E8F9 sormutató csökkentése és ellenőrzése
E8FA—E911 sormutató növelése és ellenőrzése
E912—E928 szín ellenőrzése
E929—E974 képernyőkódról ASCII kódra konvertáláshoz táblázat
E975—EAA0 SCROLL rutinok
EAA1—EB1D IRQ rutinok
EB1E—EC45 billentyűzetfigyelés generálása
EC46—EE13 billentyűzetmátrix

EE14—EEBF	üzenet a soros buszon, figyelő állapotban levő egységnek
EEC0—EEC4	másodlagos cím elküldése
EEC5—EECD	az egység figyelésének kikapcsolása, figyelő állapotba hozása után
EECE—EEE3	beszélő állapotban levő egység másodlagos címe
EEE4—EEF5	a puffer tartalma a soros buszra
EEF6—EF03	üzenet kiküldése a soros buszra (nem beszélő egység)
EF04—EF18	üzenet kiküldése a soros buszra (nem figyelő egységre)
EF19—EFA2	egy bájt behívása a soros buszról
EFA3—EFED	NMI rutin folytatása
EFEE—F035	átküldendő bájt
F036—F173	NMI rutin, mely adatokat összegyűjt RS232 részére
F174—F1E1	KERNAL üzenetek
F1E2—F1F4	üzenet kiírása a képernyőre
F1F5—F20D	karakter bevétele egy csatornáról (GET módon)
F20E—F279	karakter bevétele egy csatornáról (INPUT módon)
F27A—F2CA	kimenő karakter egy csatornára
F2C7—F308	csatorna megnyitása input részére
F309—F349	csatorna megnyitása output részére
F34A—F3EE	logikai fájl lezárása
F3EF—F3F2	az összes fájl lezárása
F3F3—F409	csatorna törlése
F40A—F541	OPEN funkció
F542—F674	LOAD RAM funkció
F675—F733	SAVE funkció
F734—F76F	idő funkció
F770—F77D	STOP billentyű lenyomásának ellenőrzése
F77E—F7AE	hibakezelő rutin
F7AF—F889	kazettafejléc keresése és olvasása
F88A—F98D	kazettaellenőrző rutin
F98E—FABC	kazettáról olvasó rutin
FABD—FBE9	bájt kezelése kazettáról olvasás közben
FBEA—FD21	kazettára író rutin
FD22—FE90	hidegindítás
FE91—FEA8	memóriaellenőrző rutin
FEA9—FF5B	NMI kezelése
FF5C—FF71	BAUD sebesség táblázat (RS232)
FF72—FF85	IRQ kezelése
FF86—FFFF	KERNAL ugrási címek (ugyanolyan az elhelyezkedésük, mint a C64-nél)

KIEGÉSZÍTÉSEK A ROM RUTINOK JOBB MEGÉRTÉSE ÉRDEKÉBEN (POKE címek)

POKE cím	Magyarázat
211	kurzorpozíció az aktuális sorban
214	az éppen aktuális képernyősorszám
97—102	akkumulátor 1.
105—110	akkumulátor 2. a számításokhoz
115—138	CHARGOT rutin
139—143	RND munkaterület
256—266	lebegőpontosból ASCII-vé konvertálási terület

(td)

ADOK–VESZEK–CSERÉLEK

Ebben a rovatunkban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közleteknek gépelt soronként (60 karakter) 100 Ft, magánszemélyeknek az első sor 50 Ft, minden további sor 20 Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

■ Sinclair Spectrum 48 k, 80 k-ra keresek angol—magyar fordítóprogramot (szótárt), széles körű felhasználói programokat, használati útmutatóval, általános iskolai 4., 5., 6. osztályos tananyaggal kapcsolatos programot. Várom azok segítségét, akik megmutatnák, hogyan lehet a játékokba „belenyúlni", hogy a feliratokat magyarra átírhassuk, a billentyű helyett pedig botkormányt használhassunk. Kékesi Lajos, Bp., Havanna u. 46. IV. em. 21. 1181. Telefon: 585-163.

■ 16 k-s ZX—Spectrumot vennék. Vastag László, Bp., Dévényi u. 18. fszt. 6. 1135. Telefon: 496-370/1708.

■ ZX—Spectrumhoz programokat cserélek. Válaszokat a programok listájával együtt erre a címre kérek: Monori Zsolt, Leninváros, Malinovszkij út 42. 3580.

■ Commodore 16-ra küldjön egy játékprogramot kazettán, és én kettőt küldök cserébe. Ha egyet kölcsönöz néhány napra, kettőt küldök vissza. Majoros Zoltán, Miskolc, Tizeshonvéd u. 18. 3525.

■ Spectrumhoz Dk'tronics fényceruza eladó. Vennék angol „Your Computer" magazinokat, 1984—85-ös számokat. Telefon: 156-411.

■ VC20 számítógép programokkal, könyvekkel együtt 8000 forintért eladó. Elek, Táncsics út 30. 5742.

■ C16, C116, Plus/4 számítógépre angol és orosz oktatóprogramok eladók. Kálmán Albert, Eger, Rákóczi út 31. III. 11. 3300. Telefonüzenet: 143-031, 330-345.

■ C64-es programokat cserélek. Kozák Zoltán, Sopron, Várfal u. 8/A, 9400.

■ C64-re készült játék- és felhasználói programokat, programleírásokat cserélek lemezen és kazettán. Pesti László, Bp., Klapka u. 3. 1193. Telefon: 780-224.

■ Eladó 3 db japán 1,8°|8,1 V|0,3 A léptetőmotor, 5 db 27 128 EPROM, gyári C16|+4 programkazetták. Vennék 74 LS 688-at. Bp. XIX., Petőfi út 1. VI. 18. Kakuk.

■ ZX—Spectrum szimulátor (repülőgép, helikopter) programokat kérek egyéb programokért. Varga István, Eger, Csapajev s. 1. III. 13. 3300.

■ C64-re készült játék- és felhasználói programot cserélek. Vukman Szabolcs, Pécs, Budai Nagy Antal u. 10. 7624.

■ ZX81 eladó. Ára 5000 forint. Telefon: 651-608.

■ ZX81 16 k-s bővítővel eladó. Árpádi János, Bp., Magyar u. 66. fszt. 6. 1212. Telefon: 486-297.

■ Lézer 310 típusú számítógép tulajdonosa vagyok. Ehhez a géphez keresek játékprogramokat. Babos Levente, Bp., Haris köz 1/A, II. 2. 1052.

■ A Mikroszámítógép Magazin 1983—85. évi számai eladók. Árajánlatot: Szeged, Postafiók 673.

■ EPROM-programozó készülékhez 28 lábas IC-foglalatok eladók (390,— Ft/db). Csányi Gábor, Mezőtúr, Ifjúsági ltp. 13. III. 16.

PRO-KONTRA GM.
1074 Budapest,
Csengery u. 7. fszt. 1/a
☎ 417-893

Cégünk ajánlataiból:

— 64 k-s memóriabővítő Commodore C16-hoz, 2300 Ft/db
— FAST—VC-1541 kommunikáció-gyorsító rendszer a C64-hez (minden gép-floppy közötti és floppyn belüli műveletet 4—12-szeresre gyorsít)

Konfiguráció beszereléssel együtt 7000 Ft

— Abszolút programvédelem C64-hez: cartridge-ben, műgyantával kiöntött EPROM-ba égetéssel, MÁSOLHATATLANNÁ teszi programjait.
— IEC buszról vezérelhető méréspontváltó egység 2 × 30 vagy 1 × 60 bemeneti, 2, ill. 4 kimeneti csatornával.
— Digitális kijelzésű elektronikus óra 8 × 14 cm-es digitmérettel, különféle színekben.
— Egyéb, közepes sorozatú fejlesztések igény szerint.

Pro-Kontra Automatizálási, Műszaki Tanácsadó és Közvetítő GM.

Budapest, Csengery u. 7. fsz. 1/a 1074
Telefon: 417-893
Levélcím: Budapest, Pf. 72. 1581
Telex: 22-7770

ZX—SPECTRUM

A Beta BASIC és alkalmazása II.

PROGRAMOK

A program használata

Betöltés: LOAD "". A program betöltése előtt NEM KELL BETÖLTENI a Beta basic 1.8-at, mert rajta van a szalagon és magától betöltődik. Tehát LOAD "" és betöltődik a loader, a Beta, a program és a szóállomány. Emiatt a töltési idő eltart egy pár percig.

Betöltődés után megjelenik az „Indítsd el a programot. (RUN+ENTER)" felirat. Ekkor a program még minden nehézség nélkül listázható, és RUN-ra elindul. RUN után megjelenik a menü.

Beírás

A beírás az alsó képernyőre történik; ENTER hatására a szöveg a memóriába és a felső képernyőre kerül. Ha a beírni kívánt szöveg már nem férne el a tárban, a felső képernyőn inverz „A szótár megtelt" felirat jelenik meg, és a program kereső üzemmódra vált.

Az elválasztó karakter a szorzásjel (CHR$ 42), ezért ezt ne használjuk a szövegben. Visszatérés a menühöz: Ø, ENTER.

A program 27Ø6Ø bájt területen tárolhat feljegyzéseket. Ezek hossza tetszőleges, de max. 255 karakter. A beírás POKE utasítással történik adott címre; ezt a program végzi el.

Keresés

A keresett szót az alsó képernyőre írva (+ENTER) megjelenik a felső képernyőn inverzben, és alatta láthatók a megtalált szópárok. Egy kérdésre több válasz is lehet, például air=affair air, aircraft, airliner, air mail stb.

A kiírás késleltetett, így ha kiírás közben megtaláljuk a kívánt szót, nyomjuk meg a BREAK-et, és visszaáll a kereső üzemmód, várva a következő keresendő szót. Ugyanez a helyzet, ha a képernyő megtelt, és megjelenik a „scroll?" felirat. Ilyenkor „n" vagy BREAK válasz esetén szintén megjelenik alul a „KERESÉS" felirat, de a felső két sor elvész.

A nagybetűvel írt, de kisbetűvel keresett szót a szótár nem találja meg. Ilyenkor, vagy ha a szó nincs a szótárban, a gép a „Nem ismerek ilyen szót" választ adja.

A keresés PAUSE 2Ø-szal lassított, de ez elhagyható. Keresésnél a program minden olyan rekordot kiír, amelyikben a keresett szó szerepel, mindegy, hogy az idegen nyelvű vagy a magyar részén.

Kimentés

A szóállomány tetszőleges néven magnóra menthető. A kimentés után kívánságra ellenőrizhető a felvétel (VERIFY). Magnótöltési hiba esetén a kimentéshez, egyébként a menühöz jutunk vissza.

Ha tévedésből nyomtuk meg a 3. billentyűt, az idézőjelből kilépve STOP utasításra visszatér a menü.

Betöltés

Ezt a pontot választva a program előbb ellenőrzi az adatmezőt, és ha az nem üres, felirat figyelmeztet annak törlésére. A fájl törlése után már elvégezhető a szóállomány betöltése. Hiba esetén a törlésre figyelmeztető felirathoz, egyébként a menühöz jutunk vissza.

Szótörlés

Ugyanúgy működik, mint a keresés, de az utoljára kiírt szópár az „l" billentyű lenyomására kitörlődik a memóriából. Más billentyűt lenyomva visszatér a szótörlés üzemmód.

Fájltörlés

A 6. billentyűt lenyomva megjelenik a „Törléshez nyomd le az ENTER-t" felirat. Az ENTER-t lenyomva törlődik az egész szóállomány, visszatér a menü alul inverz „FÁJL TÖRÖLVE" kiegészítéssel. Más billentyűt lenyomva visszajutunk a menühöz.

Információ

Megjelenik a tár mérete, a felhasznált és a szabad bájtok száma, a beírt szópárok száma és diagramon a tár telítettsége. Egy billentyűt lenyomva visszatér a menü.

A program megállítása

A sok ON ERROR utasítás hatására a program a szokott módon nem mindig állítható meg, de azért akad néhány megoldás.

1. Menünél a BREAK-et legalább 5-6 másodpercig nyomva megáll a program. Ekkor azonban, mivel az ON ERROR szubrutinként működik, de nem érkezett RETURN vagy POP utasítás, a GOSUB veremben felhalmozódnak a pr. 11. sorának címei és scroll? – "n" válasz esetén visszaugorhat a menü.

2. Meghívjuk az Információt (7 opció), ha megjelent, megnyomjuk a BREAK-et. Ekkor az alsó képernyőn megjelenik a keresés üzemmód. Most Caps shift+6, az eredmény: H STOP in INPUT, 2Ø0: 1. Ezzel a módszerrel mindig megállítható a program.

3. Az előzőből következik. Szókeresés üzemmódban minden olyan esetben, ha a vezérlés a 215 ON ERROR 2Ø0 sorról kerül vissza a 2Ø0-as sorra, a program Caps. s.+6-tal megállítható. Ez két esetben lehetséges: a scrollra BREAK (D BREAK üzenet), vagy ha a GOSUB verem üres, a 22Ø POP sor hatására (V No POP data üzenet).

Mivel az ON ERROR utasítást nem előírás szerint alkalmaztam (vagyis hibakezelő szubrutin vagy POP utasítás), előfordulhat, hogy a program megállításakor a GOSUB verem nem üres. Így megeshet, hogy listázásnál egy sroll? „n" válasz esetén visszajutunk a program egyik ON ERROR sorára. Ezért ha a program futása megállt, először CLEAR utasítást adjunk.

A program a Beta ON ERROR-ja miatt nem használ ékezetes betűket. Szintén emiatt futását az elején STOP-pal egyszer meg kellett állítani.

A program leírása

Loader

A RAMTOP-ot 27699-re állítja, kirajzolja a címlapot. Betölti a Beta basicet és meghívja (RAND. USR 58419). Eltünteti a Beta copyright feliratot (CLS), ismét kirajzolja a címlapot. Más billentyűhangot állít be (POKE 23609,5), és betölti a BASIC programot. A copyright felirat eltüntetése egyszerűbb lett volna a POKE 23624,63 (BORDER 7, INK 7) megoldással, de ekkor egy esetleges magnótöltési hiba sem lett volna olvasható.

A BASIC program (SAVE "" LINE 2)

1—5	Betölti a szóállományt (2 sor), kiírja a „RUN+ENTER" feliratot, megállítja a programot. Erre sajnos a Beta miatt van szükség, mivel az még nem jelentkezett be. Most megteheti. RUN után az első 5 sor kitörlődik (5 DELETE 1 TO 5).
1Ø—5Ø	Menü. 15 ON ERROR 15: BREAK esetén egy végtelen ciklus, ami miatt csak nehezen áll meg a program. Erre a sorra jutunk vissza a 3Ø0-as és a 4Ø5-ös INPUT sorról is STOP esetén. A GOSUB vermet minden sza-

bályos visszatérésnél a 10 CLEAR sor törli.

100—120 Adatbeírás, a beíró eljárás meghívása.

200—220 Keresés, a kereső eljárás meghívása. 215 ON ERR. 200: Scroll?-n válasz esetén visszatér a 200-as sorra. GOSUB verem törlése: 220 POP.

300—335 Kimentés. 305—310: a fájl hosszának kiszámítása. 315-ös sor: R hiba esetén újra a kimentéshez tér visszza.

400—420 Szóállomány betöltése. 400-as sor: ellenőrzi, üres-e a fájl. 410-es: R hiba esetére.

500—505 Fájltörlés. Gépi kódú rutin.

600—605 Az adatsor végét keresi. (Első 0 cím.)

700—735 Információ. 725: Szavakat számoló rutin. 730: BREAK esetén a 200-as sorra ugrik.

800—815 Szótörlés, a szótörlő eljárás meghívása. 810-es sor Sroll?-n, GOTO 800.

1000—1025 Beíró eljárás.

1005 Keresi az első szabad helyet.

1010 Ellenőrzi, van-e elég hely a beírásra. Ha nincs, figyelmeztet, és keresésre ugrik.

1015 Elválasztó karakter beírása (CHR$ 42).

1020 Beírja a szót (kiterjesztett POKE).

1100—1175 Kereső eljárás.

1105 C=a keresett szó első előfordulási helye 28700—55760 között.

1110 Ha C=0, nincs ilyen szó, az eljárás végére ugrik.

1115 h: a szótörlő eljárás miatt felvett változó. Értéke 1, ha a program megtalálta a keresett szót, 0, ha nem.

1120—1140 C-től visszafelé keresi az első elválasztó karaktert. Értéke C1 lesz.

1145 C1-től előre keresi az első elválasztó karaktert. Értéke C2.

1150 Ha nem találja, a keresett szó az utolsó a fájlban.

1155—1160 Kiírja a két elválasztó karakter közötti teljes szöveget. 1160 PAUSE 20: a kiírás késleltetése. Ez elhagyható, de így barátságosabb a keresés.

1165 C2-től keresi a következő szót.

1170 Ha C=0, nincs több ilyen szó, eljárás vége. Ha nem, visszatér az elválasztó karakterek kereséséhez.

1200—1245 Szótörlő eljárás.

1205 A kereső eljárás meghívása. Ha nincs ilyen szó (h=/0), az eljárás végére ugrik.

1210—1215 Ha nem "1"-et nyomsz, az eljárás végére ugrik.

1225 Megkeresi a fájl végét (GOSUB 600), C1—C2 között kitörli a mezőt.

1230 C1-re tölti a törölt szó mögötti részt.

1235—1240 A fájl végéről kitörli a maradékot, és tájékoztat, hogy a szótörlés megtörtént.

Gépi kódú rutinok

Karakterszámláló rutin

```
                        ORG   55760
D9D0   01B469           LD    BC,27060 ; ciklusszám
D9D3   211C70           LD    HL,28700 ; kezdőcím
D9D6   110000           LD    DE,0     ; itt lesz az eredmény
D9D9   3E2A      L1     LD    A,42     ; CHR$ 42
D9DB   EDA1            CPI            ; összehasonlít
D9DD   2001             JR    NZ,L2    ; ha nem egyezik, ugrás
                                       L2-re
D9DF   13               INC   DE       ; ha igen, DE-t növeli
D9E0   78        L2     LD    A,B      ; B-t A-ba tölti
D9E1   B1               OR    C        ; és összehasonlítja C-vel
D9E2   20F5             JR    NZ,L1    ; ha nem egyezik, ugrás
                                       L1-re
D9E4   D5        PUSH DE
D9E5   C1               POP   BC       ; eredményt a BC-be
D9E6   C9               RET
```

Vagy DATA 1,180,105,33,28,112,17,0,0,62,42,237,161,32,1,19,120, 177,32,245,213,193,201

Fájltörlő rutin

```
                        ORG   55783
D9E7   1600             LD    D,00     ; 0-t tölt, majd az adott
                                       c.
D9E9   211C70           LD    HL,28700 ; kezdőcím
D9EC   01B469           LD    BC,27060 ; ciklusszám
D9EF   72        CIKL LD    (HL),D   ; beírja a 0-t
D9F0   23               INC   HL       ; növeli a HL-t
D9F1   0B               DEC   BC       ; csökkenti BC-t
D9F2   78               LD    A,B,     ; A-ba tölti B-t
D9F3   B1               OR    C        ; és összehasonlítja C-vel
D9F4   20F9             JR    NZ,CIKL  ; ha nem egyezik, ugrás
                                       CIKL-ra
D9F6   C9               RET            ;ha igen, vége
```

Vagy DATA 22,0,33,28,112,1,180,105,114,35,11,120,177,32,249,201.

Az első rutin elhagyható, a második Beta basic-kel helyettesíthető. Például CLEAR 34799. A fájlt 21000 bájtra csökkentjük. A program 500-as sorát kicseréljük a következőre:

500 FOR n= 34800 TO 52300 STEP 3500: POKE n, STRING$(3500,CHR$0):NEXT n.

Ez a sor a 34800-as címtől kezdve 6 db 3500 karakter hosszúságú, 0-ból álló sztringet ír a memóriába, egészen az 55800-as címig. Ezenkívül még a következő javításokra van szükség: ha a Beát külön szalagról töltjük be, elmarad a loader, a program eleje pedig így módosul:

1 LOAD "" CODE : REM szóállomány betöltése
2 DELETE 1 TO 2 . Ekkor a program autostarttal indulhat.

Át kell javítani még a következő sorokat:

310 SAVE m$ CODE 28700, cím—28700
400 IF cím≠ 28700
600 LET cím=INSTRING/28700,MEMORY$..

ezekben a sorokban a 28700-at 34800-ra kell javítani.

700 sor a 27060-at 21000-re,
a 28700-at 34800-ra,
az 55760-at 55800-ra

705—720 sor elmaradhat (ez a diagram)

725 sor törölve (karaktersz. rutin hívása)

1010 sor 55759-et 55799-re

1105 sor 28700-at 34800-ra
55760-at 55800-ra

1145, 1165 sorok
55760-at 55800-ra kell javítani

Természetesen más értékek is kipróbálhatók. A lényeg az, hogy a fájl törlése még elfogadható sebességgel történjen, és legyen elég hely a RAMTOP alatt a STRING$ függvény számára.

A program elhelyezkedése a memóriában

RAMTOP:	28699
Szóállomány:	28700—55759 27060
Karaktersz. rutin:	55760—55782
Fájltörlő rutin:	55783—55798
Beta basic:	55801—65367
UDG: üres	

A program és a RAMTOP közötti szabad terület CLEAR után
Az 1—5. sor kitörlése előtt: 556 bájt
Az 1—5. sor kitörlése után: 724 bájt

A szóállomány

Jelenleg angol—magyar szavak vannak a szótárban, számuk 789. Ez 16949 bájt területet foglal le; szabad még 10111 bájt. Ha a program futását megállítjuk és kiadjuk a következő parancsokat: GOSUB 600: PRINT MEMORY$ // (28700 TO cím), láthatjuk, hogy ez 24 sűrűn teleírt képernyőt jelent. Ennek ellenére a program egy oxfordi szótárral nem tudja felvenni a versenyt. Egy angol—magyar miniszótárban kb. 4000, az Országh László-féle kiszszótárban közel 22 000 szó található. Mint már említettem, a rendelkezésre álló 27 060 bájt területre a hosszúságtól függően nagyjából 1000—1200 szópár írható be. Ezért ajánlatos csak a nem ismert szavakat beírni, és megtanulás után kitörölni.

A mostani szóállomány a következő képet mutatja. Először elkezdtem bemásolni az angol—magyar miniszótár szavait. Eljutottam a c betű elejéig, és akkor rájöttem, hogy elvetettem a sulykot: ez kb. 7000 bájt. Idáig a szavak rendezettek. Ezek után egy angol könyvet olvasva csak a nem ismert szavakat szótáraztam ki. Innen kezdve rendezetlen az állomány, de a visszakeresés szempontjából szerintem ennek nincs jelentősége. Minden szót kisbetűvel írtam be, így a gép „C" kurzorral semmit nem talál meg.

Még egy szépséghiba: a szótár nem használja az ékezetes betűket. Megoldható lett volna grafikus karakterekkel, ekkor azonban a kiírás idejére ki kellett volna kapcsolni a Beta basicet KEYWORDS 0 utasítással. Ilyenkor előfordulhatott volna, hogy a kiírás közben BREAK-et nyomva leáll a program. A Beta kulcsszavak ugyanis grafikus üzemmódban érhetők el KEYWORDS 1 esetén.

LITAUSZKY GYÖRGY

Manó a Spectrumon

A Commodore-hívők egyik érve a Spectrummal szemben: a Sinclair-gép nem tud sprite-okat kezelni. A Spectrumot a hardver oldaláról valóban elég szegényen látta el Sir Clive. Programmal azonban — ha mindent nem is —, sok érdekes dolgot el lehet érni! Az alábbiakban egy 16×16-os, hátteret nem takaró, nem nagyítható, nem színes, szóval elég buta manót szimuláló programot mutatok be.

Az alapelv

A Spectrumban levő ULA másodpercenként 50 alkalommal megszakításjelet küld a Z80-nak. A processzor erre — az IM 1 módban — a 0038H címre ugrik; továbbírja a Spectrum óráját és elvégzi a billentyűzet dekódolását, majd folytatja, amit félbehagyott. Saját megszakításrutinokat, amelyek 20 ms-onként lefutnak, a Z80 IM 2 módjának felhasználásával készíthetünk. Az elég bonyolult IM 2 megértését is segíti, ha most végigkövetjük, hogyan működik a manó.

A megszakítás pillanatában a Z80 beolvassa az adatbuszon levő bájtot egy munkaregiszterbe. Ez a regiszter (alsó bájt) az I regiszterrel (felső bájt) együtt egy 16 bites memóriacímet ad. Erről a memóriacímről olvassa ki a megszakításrutin kezdőcímét a központi egység. A Spectrum sajnos nem használja ki az ebben az ugrótáblázatban rejlő lehetőségeket, terheletlen adatbuszról FFH-t olvas be alsó bájtnak.

A manórutin is egy megszakításprogram. Megvizsgálja, változott-e a manó pozíciója az elmúlt 0,02 másodperc során. Ha igen, törli a manót régi helyéről és kirakja az újra. Egyszerű feladat, csakhogy igen gyorsan kell megcsinálni: amíg az elektronsugár a keret felső szélét rajzolja. Mert ha nem készülünk el időben, és a manó épp a képernyő tetején van, csúnya villogást keltünk.

A minél nagyobb sebesség érdekében korlátoztam a manó tulajdonságait a fentebb felsoroltak szerint.

A program sajnos még így sem elég gyors: ha a manót a képernyő felső széle közelében 3, 4 vagy 5 pixelenként mozgatjuk jobbra vagy balra, a manó alsó része villog! Ez ellen sajnos csak megfelelő programozással védekezhetünk.

Használat

A Spectrum igen nehezen engedi meg új utasítások definiálását, ezért a manót csak a C64 alapgép módszerével: POKE-okkal vezérelhetjük.

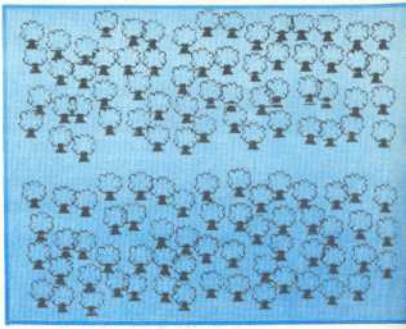
65 297: x koordináta
0 .. 255

65 298: y koordináta
0 .. 255

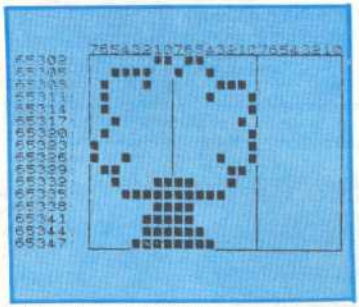
a bal
felső
pixel
helye

Ezek a koordináták nem egyeznek meg a Spectrum grafikáéival, mert a 0,0 pont a képernyő bal *felső* sarkát jelöli. Megfelelő koordináták megadásával elérhetjük, hogy a manó fent, jobbszélen vagy lent „kiúszszon" a képből. Bal szélen erre a mutatványra nem képes.

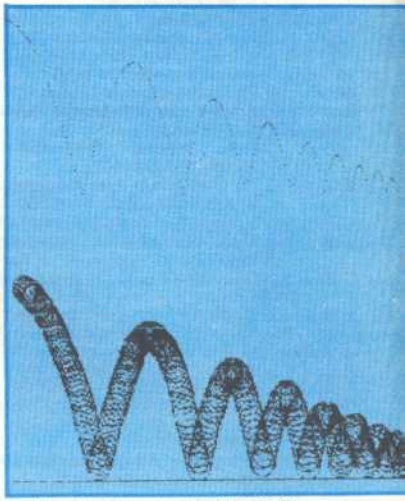
Mozgatásnál vegyük figyelembe, hogy bármilyen gyorsan is írjuk az új koordinátákat, a manó csak 50-ed másodpercenként ugrál. Gépi kódú programból azt is megvizsgálhatjuk: ugrott-e már a manó. A 65 295 (xØ) és 65 296 (yØ) változók a manó tényleges pozícióját mutatják.



2. ábra. Az sp-erdő futási eredménye



1. ábra. A manó térbeli képe



3. ábra. A labdapattogtatás két ábrája

1. lista

```
100 REM ** erdőrajzolás **
110 REM
120 REM
130 LET xx=65297: LET yy=xx+1
140 LET mod=65226
150 LET sp=65302
160 REM
170 RESTORE 600: GO SUB 400: CL
S
175 POKE mod,182
180 FOR x=5 TO 240 STEP 16
185 LET n=1
190 FOR y=5+INT (12*RND) TO 160
STEP 18
195 LET n=n+1: IF n=6 THEN GO T
O 230
200 LET i=x+INT (3-7*RND)
210 LET j=y+INT (3-7*RND)
220 PAUSE 1: POKE xx,i: POKE yy
,j
230 NEXT y
240 NEXT x
398 POKE yy,192: POKE xx,0
399 STOP
400 POKE xx,0: POKE yy,192: PAU
SE 1
410 FOR i=0 TO 47
420 READ byte: POKE sp+i,byte
425 POKE mod,174
430 NEXT i
440 RETURN
600 REM ** kerek fa **
610 DATA   3, 96,0
611 DATA  60,144,0
612 DATA  66, 14,0
613 DATA  64, 17,0
614 DATA  64,  1,0
615 DATA  32,  2,0
616 DATA  64,  1,0
617 DATA 128,  1,0
618 DATA 144,  9,0
619 DATA  96,  6,0
620 DATA  35,196,0
621 DATA  31,248,0
622 DATA   3,192,0
623 DATA   7,192,0
624 DATA   7,224,0
625 DATA  15,240,0
800 BORDER 0: PAPER 0: INK 9
810 CLEAR 65022
820 LOAD "sp_code"CODE
830 RANDOMIZE USR 65351
840 GO TO 100
900 SAVE "sp_erdo" LINE 800
910 SAVE "sp_code"CODE 65023,33
0
920 VERIFY "sp_erdo"
930 VERIFY "sp_code"CODE
940 STOP
```

2. lista

```
100 REM ** labd **
102 CLEAR 65022: LOAD ""CODE
110 LET xx=65297: LET yy=xx+1
REM koordinatak
120 LET sp=65302: REM sprite be
zo lefoglalas
130 LET mod=65226: REM XOR/OR b
ol
138 REM
140 REM ** palya kirajzolasa **
145 REM
150 DIM p(256): FOR i=1 TO 256
160 LET f=(i-1)/(350-i)*10
170 LET p(i)=160-ABS INT (100*C
OS (i*EXP (-f/10)))
180 PLOT i-1,160-p(i): NEXT i
190 GO SUB 600: CLS
200 RESTORE 500: GO SUB 400: RE
M labda definialasa
210 PLOT 0,0: DRAW 255,0
220 FOR i=1 TO 255: POKE xx,i:
POKE yy,p(i): NEXT i
230 POKE yy,192: POKE xx,0
240 RESTORE 550: GO SUB 400: RE
M kor definialasa
250 GO SUB 600
260 POKE mod,182: REM OR mod
270 FOR i=1 TO 255: POKE xx,i:
POKE yy,p(i): NEXT i
280 POKE mod,174: REM XOR mod
290 GO SUB 600: CLS
300 GO TO 200
400 REM ** definialo rutin **
410 POKE xx,0: POKE yy,192: PAU
SE 1
420 FOR i=0 TO 47
430 READ byte: POKE sp+i,byte
440 NEXT i
450 RETURN
500 REM ** labda **
510 DATA   7,224,0
511 DATA  24, 24,0
512 DATA  33,132,0
513 DATA  66,130,0
514 DATA  66,  6,0
515 DATA 153, 66,0
516 DATA 126, 49,0
```

```
517 DATA 131,   1,0
518 DATA 179,   1,0
519 DATA 176,   7,0
520 DATA 128,103,0
521 DATA  64,  98,0
522 DATA  76,   2,0
523 DATA  44,   4,0
524 DATA  24,120,0
525 DATA   7,224,0
550 REM **  kor  **
560 DATA   7,224,0
561 DATA  24,  24,0
562 DATA  32,   4,0
563 DATA  64,   2,0
564 DATA  64,   2,0
565 DATA 128,   1,0
566 DATA 128,   1,0
567 DATA 128,   1,0
568 DATA 128,   1,0
569 DATA 128,   1,0
570 DATA 128,   1,0
571 DATA  64,   2,0
572 DATA  64,   2,0
573 DATA  32,   4,0
574 DATA  24,  24,0
575 DATA   7,224,0
600 PRINT #1;AT 1,0;"Nyomj le e
gy gombot !           ": PAUSE
0
610 PRINT #1;AT 1,0;"
                      ": RETURN

900 SAVE "sp_demo" LINE 100
910 VERIFY ""
```

```
  1 DEF FN a$(i)=CHR$ (i+48+7*(
i>9))
  2 DEF FN b$(i)=FN a$(INT (i/1
6))+FN a$(i-16*INT (i/16))
  3 DEF FN c$(i)=FN b$(INT (i/2
56))+FN b$(i-256*INT (i/256))
  4 DEF FN a(i$)=CODE i$-48-7*(
i$(1)>"9")
  5 DEF FN b(i$)=FN a(i$(1))*16
+FN a(i$(2))
  6 DEF FN c(i$)=FN b(i$(1 TO 2
))*256+FN b(i$(3 TO 4))
 10 CLEAR 65022
100 DIM a$(27)
110 FOR i=65023 TO 65366 STEP 8
120 INPUT (FN c$(i);": ")'a$
130 LET s=0
140 FOR j=0 TO 7
150 LET c=FN b(a$(3*j+1 TO 3*j+
2))
160 LET s=s+c
170 POKE i+j,c
180 NEXT j
190 LET st=FN c("0"+a$(25 TO ))
200 IF s<>st THEN BEEP 2,2: PRI
NT a$: GO TO 120
210 PRINT FN c$(i);" ok."
220 NEXT i
230 PRINT "kész"
240 SAVE "sp_code"CODE 65023,33
8
250 VERIFY ""CODE
```

3. lista

4. lista

```
FDFF 01 FE F5 E5 D5 C5 DD E5,635
FE07 21 08 FF E5 2A 11 FF ED,434
FE0F 56 0F FF B7 ED 52 C8 78,4A2
FE17 F5 CD 8D FE 2A 11 FF 22,4A9
FE1F 0F FF 7C 0F 0F 0F E6 1F,280
FE27 DD 6F F1 57 7D 94 D2 62,4E9
FE2F FE ED 44 E6 07 28 5C FE,49E
FE37 05 DA 6E FE ED 44 C6 08,44A
FE3F DD 67 21 16 FF 06 30 B7,367
FE47 CB 1E 23 10 F8 3A 13 FF,363
FE4F 3C FE 08 21 16 FF CC EE,432
FE57 FE 32 13 FF DD 25 C2 44,44A
FE5F FE 18 30 E6 07 28 2C FE,388
FE67 05 38 D5 ED 44 C6 08 DD,3EE
FE6F 67 21 45 FF 3A 13 FF B7,3CF
FE77 CC FB FE 30 32 13 FF 06,440
FE7F 30 B7 CB 16 28 10 F8 CD,3DB
FE87 2A C2 70 FE 18 05 3A 14,2C0
FE8F FF DD 6F CD 7D 32 14 FF,4EA
FE97 3A 10 FF DD 67 0E 11 81,2CD
FE9F 16 FF DD 7C 5F B7 1F 37,3DA
FEA7 1F B7 1F AB E6 F8 AB 57,430
FEAF 7B 17 17 E6 F0 5F 0D C8,3A3
FEB7 7A FF 53 C2 E8 FE DD 70,5E2
FEBF B3 5F 06 03 DD 70 FE 20,393
FEC7 30 03 1A AE 12 DD 2C 13,229
FECF 23 10 F1 DD 2D DD 2D DD,415
FED7 2D DD 24 DD 7C E6 07 CA,43E
FEDF A1 FE 18 18 18 14 C3 B5,37C
FEE7 FE 23 23 23 C3 C8 FE D6,4D6
FEEF 08 E5 54 5D 23 01 30 00,1F2
FEF7 ED B0 E1 C9 C6 08 E5 54,54E
FEFF 5D 28 01 30 00 ED B8 E1,33F
FF07 C9 DD E1 C1 C1 C3 3A 00,516
FF0F 00 C0 00 C0 00 00 00 FF,27F
FF17 FF 00 80 01 00 80 01 00,201
FF1F 80 01 00 80 01 00 80 01,183
FF27 00 80 01 00 80 01 00 80,182
FF2F 01 00 80 01 00 80 01 00,103
FF37 80 01 00 80 01 00 80 01,183
FF3F 00 80 01 00 FF FF 00 00,27F
FF47 F3 3E FD ED 47 ED 5E FB,5A8
FF4F C9 FE 02 02 02 02 02 02,103
```

Amikor a manó a képernyőn van, ne töröljük le felelőtlenül, mert a csalós törlési módszer miatt egy halott árnyképet vagy „szemetet" hagyhat maga után!

Definiálás

A minél nagyobb sebesség érdekében a 16×16-os pontmátrix egy 48 bájtos mezőben helyezkedik el, néhány bittel eltolva. Hogy pontosan hol van, azt csak akkor tudjuk megmondani, ha x koordinátáját Ø-ra állítjuk. Ekkor a manó helyzete az 1. ábra szerinti. Szintén a XOR törlési módszer miatt definiáláskor a manó nem lehet a képernyőn, mert „szemetet" hagy maga mögött.

A manóelhelyező szubrutin közepében van egy XOR (HL) utasítás. Ha ezt OR (HL)-re javítjuk, a manóemulátor program manósokszorozóvá válik, mert nem fogja törölni saját régi képeit. Ennek felhasználására mutat példát az *1. lista*, az erdőprogram.

Ennyi kikötés után talán már nem is lehet használni a programot — gondolhatják. Egyszerű ellenpélda a *2. listán* látható program, amely egy pattogó labdát rajzol.

Végül az egyik legfontosabb dolog: beolvasás után aktiválni kell az emulátort a 65 351 címen levő szubrutin hívásával (I regiszter feltöltése, IM 2 mód beállítása). Ilyenkor a manó egy 16 pixel oldalú üres négyzet; pozíciója (Ø,192), azaz a képernyő bal alsó sarka alatt van, nem látható.

A program beírása a *3. listán* látható segédprogrammal történik. Soronként begépeljük a *4. listán* látható hexadecimális számokat (az ellenőrző összegeket is), ha szükséges, a hibákat kijavítjuk, majd a programot kipróbálás előtt magnóra vesszük.

Ha valakit nagyon érdekel a program működése, levélben elküldöm a megjegyzésekkel ellátott, kb. 200 soros forráslistát. Mivel a programban az XL, XH regiszterek is előfordulnak, a legtöbb disassembler csütörtököt mondana a visszafejtésnél.

A manókezelő program adatai

65 023: ugrótáblázat a megszakításrutinhoz
65 025: a megszakításrutin belépési pontja
65 295: xØ a manó pillanatnyi (régi) helyzete
65 296: yØ a manó pillanatnyi (régi) helyzete
65 297: x a manó következő (új) helyzete
65 298: y a manó következő (új) helyzete
65 299: pos = x mod 8; a manó helyzete a 48 bájtos mezőben
65 300: xt = x/8; a manó koordinátája karakterpozíciókban
65 302: sp a manó 48 bájtos mezőjének kezdete
65 351: act az aktivizáló rutin belépési pontja
65 359: end az utolsó foglalt bájt

A módosítható jellemzők

65 209: a képernyő alsó szélének vizsgálata. Alapértéke 88. Ha 80-ra javítjuk, a manó csak a felső kétharmadban, ha 72-re, csak a legfelső harmadban látható.
65 222: a képernyő jobb szélének vizsgálata; értéke 32.
Ha csökkentjük, a manó mozgásterét jobbról 8 pontonként korlátozzuk.
65 229: a manóemulátor/manósokszorozó módot meghatározó utasítás címe. Értéke 174 (= XOR (HL)), átírható 182-re (= OR (HL)).

MÉSZÁROS LÁSZLÓ

PTA—4000

Új karakterek

A PTA—4000 gépen lehetőség van arra, hogy a nem használt 128—255 ASC kódoknak karaktereket definiáljunk: a 785D h bájtot kell Ø-ra, a 785E h bájtot pedig a bővítés címére átírnunk. Ez utóbbi csak a cím H bájtot jelenti, az L bájt ugyanis kötelezően Ø. A bővítés főbb részei:

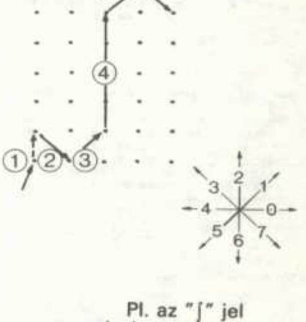
1. *Képernyőkarakterek* területe. Kezdőcíme az általunk meghatározott BH ØØ. (BH-ra állítottuk be a 785E h bájtot.) Egy karaktert 5 bájt ír le, egy bájt GPRINT-nek megfelelően lesz értelmezve.

2. *Printer-címmutató.* A B+2H ØØ, B+2H Ø1 bájtokon található kétbájtos cím, mely a 3. első bájtját adja.

3. *Printer-címtábla.* A 2. által mutatott címen kezdődik, kátbájtos címeket tartalmaz, melyek a karakterek adatainak első bájtjára mutatnak. Az első cím a 128 kódú karakter első bájtjára mutat.

4. *Printer-karakterek.* Minden karakter a 3-ban neki megfeleltetett címen kezdődik. Egy karaktert egy bájtsorozat ad meg. Egy bájt értelmezését az *ábra* mutatja.

A rajz mindig a bal alsó sarokból kezdő-



Pl. az "∫" jel

	7	6	5	4	3	2	1	Ø
1	Ø	Ø	Ø	1	Ø	Ø	Ø	1
2	Ø	1	1	1	1	Ø	Ø	1
3	Ø	1	Ø	Ø	1	Ø	Ø	1
4	Ø	1	Ø	1	Ø	1	Ø	Ø
5	Ø	1	Ø	Ø	1	Ø	Ø	1
6	1	1	1	1	1	Ø	Ø	1

Ø–2 bit hossz (>Ø!)
3–5 bit irány

6 bit toll le = 1, fel = Ø
7 bit rajz vége = 1

```
DISPLAY KARAKTEREK

>> 
4300 > 20 14 12 14 20 38 40 40
4308 > 38 40 2E 52 57 52 42 38
4310 > 54 57 54 18 3C 42 42 42
4318 > 3C 38 44 42 44 38 3D 42
4320 > 42 42 3D 38 45 44 45 38
4328 > 3C 43 42 43 3C 38 42 44
4330 > 42 38 3E 40 43 40 3E 3C
4338 > 43 40 3C 40 3E 41 40 41
4340 > 3E 3D 40 41 3C 40 3E 43
4348 > 40 43 3E 3F 40 43 3C 40
4350 > 20 40 3E 01 02 20 40 38
4358 > 05 02 04 02 2F 02 04 18
4360 > 28 2F 28 18 00 1C 2A 00
4368 > 00 00 00 2A 1C 00 14 22
4370 > 2F 22 14 12 09 12 24 12
4378 > 00 04 02 00 00 04 02 04
4380 > 03 00 14 00 14 14 14 03
4388 > 3B 44 44 20 40 40 2F 40
4390 > 40 2F 40 40 20 40 20 38
4398 > 20 34 22 14 34 1C 16 14
43A0 > 60 50 40 44 2E 3E 41 5D
43A8 > 41 3E 63 55 49 41 63 07
43B0 > 40 2F 40 07 5C 62 02 62
43B8 > 5C 30 40 30 2E 41 2F 01
43C0 > 25 2A 10 22 51 3F 04 02
43C8 > 38 40 4A 4D 35 00 15 52
43D0 > 55 22 01 06 01 01 2E 44
43D8 > 30 10 30 44 40 22 11 3E
43E0 > 40 4E 55 29 05 00 2F 00
43E8 > 00 00 00 04 4A 3F 0A 04
43F0 > 2F 49 45 45 32 2F 01 01
43F8 > 01 02 20 2E 21 3F 60 63
4400 > 1C 2F 1C 63 2F 20 1C 02
4408 > 2F 2E 21 1A 05 2E 40 3E
4410 > 01 01 2F 3F 20 20 3F 40
4418 > 02 00 00 04 2F 2F 40 2F
4420 > 40 3F 20 2F 02 2E 40 40
4428 > 30 41 49 49 2A 1C 2F 00
4430 > 3E 41 3E 46 29 19 09 2F

PRINTER CIMTÁBLA

>> 
4440 > 02 02 02 00 02 17 02 28
4448 > 02 2C 02 38 02 44 02 51
4450 > 02 5E 02 68 02 70 02 02
4458 > 02 8E 02 99 02 00 02 B1
4460 > 02 BE 02 C4 02 D3 02 D8
4468 > 02 DD 02 E2 02 E7 02 EF
4470 > 02 F7 02 FB 03 01 03 05
4478 > 03 15 03 18 03 22 03 28
4480 > 03 31 03 35 03 44 03 4D
4488 > 03 57 03 62 03 6B 03 75
4490 > 03 7E 03 8B 03 96 03 9E
4498 > 03 A5 03 AB 03 B2 03 BF
44A0 > 03 CC 03 D5 03 D0 03 E1
44A8 > 03 F2 02 F8 04 01 04 06
44B0 > 04 11 04 18 04 1E 04 27
44B8 > 04 30 04 38 04 42 04 4D

GÉPI KÓDO PROGRAMBETÉT

>> 
44C0 > FD 58 FD 5A FD 52 4A 48
44C8 > 6A 3F 94 F9 83 41 44 00
44D0 > 87 FD 48 4A 00 34 F9 83
44D8 > 41 34 6E 28 5E E9 28 5D
44E0 > 00 FD 58 5A C6 85 38 6A
44E8 > 12 51 00 83 E9 28 75 00
44F0 > A4 DD DD DD 28 85 18 6A
44F8 > 55 CD 92 00 00 CD AC 9A

PRINTER CIMMUTATÓ

>> 
4500 > 01 48
```

```
PRINTER KARAKTEREK

>> 
4502 > 54 4A 2A 13 22 31 24 12
450A > E4 82 61 59 52 48 42 23
4512 > 29 15 69 12 F2 56 44 36
451A > 64 13 43 14 21 F2 83 62
4522 > 55 52 48 42 25 21 64 15
452A > 82 F2 82 81 59 54 45 42
4532 > 29 24 65 61 12 F2 82 61
453A > 59 52 49 42 29 22 65 61
4542 > 12 F2 12 41 82 41 32 24
454A > 69 62 58 54 49 42 F5 16
4552 > 41 82 41 33 22 68 62 59
455A > 52 49 42 F9 81 58 54 49
4562 > 42 29 24 69 62 15 52 82
456A > F2 81 58 52 49 42 29 22
4572 > 69 62 15 52 82 F2 81 59
457A > 55 82 11 22 82 11 25 69
4582 > E2 82 61 58 53 83 23 29
458A > 19 69 12 F2 11 55 11 41
4592 > 82 41 31 25 69 62 D8 11
459A > 53 12 41 81 41 32 23 29
45A2 > 15 69 61 D9 81 58 55 84
45AA > 25 69 62 12 22 8A F2 11
45B2 > 53 11 52 82 22 38 23 29
45BA > 15 69 61 D8 11 29 49 54
45C2 > 49 F9 11 29 49 51 61 59
45CA > 49 42 28 68 61 52 51 49
45D2 > F9 82 56 2A 1A EA 82 5A
45DA > 84 6A D6 13 4A 3A 64 FA
45E2 > 13 44 5A 3A EA 82 56 2A
45EA > 32 6A 5A 12 CA 12 49 2A
45F2 > 49 12 69 5A E9 16 82 22
45FA > E9 13 49 52 82 22 E8 11
4602 > 41 81 42 12 82 21 E1 14
460A > 42 52 62 22 82 69 22
4612 > 29 42 C9 44 22 D6 44 21
461A > 52 59 62 39 29 1A 13 F6
4622 > 11 44 14 6C 31 83 52 59
462A > E1 8C 64 32 44 13 EC 44
4632 > 55 6C F1 81 59 54 49 42
463A > 29 24 69 62 12 52 82 22
4642 > 11 E2 44 51 14 51 64 21
464A > 2A 6A F1 16 22 29 42 49
4652 > 52 22 26 21 C2 41 51 59
465A > 52 49 42 29 29 69 61 C1
4662 > 81 59 51 49 2B 23 49 52
466A > C9 56 42 29 69 61 42 29
4672 > 21 69 E2 81 41 49 55 61
467A > 69 35 59 CC 8B 5A 49 41
4682 > 33 22 69 61 59 51 49 41
468A > F9 82 41 49 59 62 59 51
4692 > 4A 29 69 DA 84 56 61 69
469A > 21 51 59 E9 4C 24 29 22
46A2 > 8A 22 F9 4A 2A 1A 53 59
46AA > E9 41 49 5A 51 29 41 49
46B2 > 51 59 61 69 F1 56 83 22
46BA > 69 61 59 83 F9 11 29 49
46C2 > 55 29 69 21 29 42 49 51
46CA > 59 E2 43 49 51 59 63 33
46D2 > 56 43 F9 56 43 F9 52 44
46DA > 22 12 59 53 62 23 E9 51
46E2 > 49 52 58 51 84 21 69 22
46EA > 29 21 22 52 61 82 61 D3
46F2 > 4A 52 4A 26 24 D6 4A 52
46FA > 4A 26 24 56 89 29 C9 49
4702 > 54 49 42 F6 82 61 59 55
470A > 83 25 69 89 29 21 D1 84
4712 > 56 32 69 62 58 D2 44 56
471A > 22 26 22 D6 44 41 21 13
4722 > 56 22 26 22 D6 55 84 25
472A > 24 42 49 51 59 E2 15 41
4732 > 25 42 49 51 55 E2 11 29
473A > 42 49 54 59 62 69 3A C2
4742 > 56 83 29 24 69 59 52 22
474A > 42 52 C9 48 41 63 59 51
4752 > 49 43 F6

BEJELENTKEZŐ SZÖVEG

>> 
4755 > 42 89 26 69 24 83 23 28
4750 > 62 65 24 87 6C 24 26 65
```

A bővítéssel kapott új karakterek és ASC kódjaik :

CHR	ASC	CHR	ASC
Aa	128	àa	129
Ee	130	èe	131
Oo	132	òo	133
Oo	134	öo	135
Oo	136	õo	137
Uo	138	úu	139
Uo	140	üu	141
Oo	142	üu	143
ſſ	144	¢¢	145
↑↑	146	↓↓	147
←←	148	→→	149
↕↕	150	≈≈	151
''	152	""	153
==	154	ee	155
⊥⊥	156	⊨⊨	157
≮≮	158	≠≠	159
ΔΔ	160	ΘΘ	161
ΣΣ	162	ΨΨ	163
ΩΩ	164	∂∂	165
ββ	166	∂∂	167
δδ	168	ζζ	169
ηη	170	××	171
λλ	172	ρρ	173
ΗΗ	174	ΦΦ	175
Бб	176	Гг	177
Дд	178	Жж	179
Ии	180	Йй	181
Лл	182	Цц	183
Чч	184	Шш	185
Щщ	186	Ыы	187
Ьь	188	Ээ	189
Юю	190	Яя	191

```
ÁáÉèÓóÖöÕõÚúÜüÙùſ¢
↑↓←→↕≈'"=£⊥⊨≮≠ΔΘΣΨ
ΩαβδζηλρΗΦБГДЖИЙ
ЛЦЧШЩЬЪЭЮЯ
```

dik. Egy karakter tetszőlegesen nagy lehet, de a tollnak a rajz végén mindig egy normál karakteren belül kell lennie.

A bővítést POKE utasítással írjuk be a 4300h, 4764h területre, majd rögzítsük kazettára:

CSAVE M „karakterek"; & 4300, & 4764

A kazettáról most már betölthetjük egy tetszőleges BH 00 című területre: B=... (a terület H bájtja).

CLOAD M B*256, ezután

CALL B*256+448 meghívjuk a beépített programot, mely a következőket végzi el:

— a printer-cimtábla és printer-cimmutató relatív címeihez hozzáadja P regiszter H bájtját, így előállítja az abszolút címeket;

— kitörli az 1. programrészt, mert ha újból meghívjuk a programot (minden bekapcsolás után meg kell hívni), a címtábla és címmutató területeken már abszolút és nem relatív címek vannak;

— beállítja a 785D h és 785E h címeket (kikapcsolás után 785D h címre FF h kerül, ezért bekapcsolás után vagy át kell írni, vagy meghívni a segédprogramot);

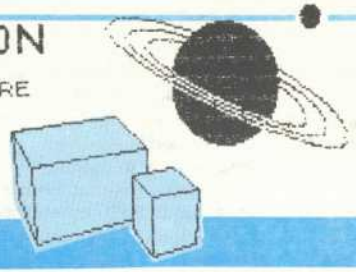
— ha minden rendben volt, a „Bővítés betöltve" szöveget kb. három másodpercig megjeleníti a képernyőn.

A bővítéssel csak 128—191 kódú karaktereket kaptunk. 191-nél nagyobb kódú karaktert ne használjunk, mert hibás működést eredményez! A bővítés utolsó bájtja B*256+1124. Célszerű a bővítést a memória legelejére tölteni, és egy „NEW B*256+1125"-tel letiltani a BASIC elől!

Vigyázzunk a RESERVE területre. Ezt nagyon jól használhatjuk: például az egyes billentyűknek az ékezetes betűk jelentést adhatjuk, a fennmaradók egyikének pedig a „POKE&785D,Ø a" jelentést, mely ki- és bekapcsolás után visszahozza a bővítést.

MÉSZÁROS CSABA

PSION
SOFTWARE



VU-3D rajzolóprogram

A VU—3D-t 1982-ben készítette a PSION szoftvercég. A program 48 k-s Spectrum gépen fut, háromdimenziós tárgyak rajzolására és ábrázolására alkalmas.

Az ábrázolt testek teljesen körbeforgathatók a képernyőn, így minden irányból megvizsgálhatjuk azokat. A képet nagyíthatjuk és kicsinyíthetjük, közelíthetjük és távolíthatjuk. A közelítés addig fokozható, hogy a képernyővel, mint egy szeletelő síkkal, metszeteket is készíthetünk a rajzról. A pillanatnyi képet bármikor ki lehet mereviteni. Ilyenkor a test tetszőleges irányból beárnyékolható, vagy a nem látszó élek kitörölhetőek. Az így kapott ábrát kinyomtathatjuk vagy screenként magnószalagra menthetjük.

Figyelembe véve a gép lehetőségeit, azt nem állíthatom, hogy a programmal alkatrészek is tervezhetők, de mértan, ábrázoló geometria stb. tanulásához hasznos segítséget nyújthat, és a térlátást is fejleszti. Azonkívül szórakozásnak sem utolsó egy-egy bonyolultabb alakzatot több irányból is megnézni.

A program elég elterjedt, de leírás hiányában kevesen vannak tisztában minden funkciójával. Mivel nekem már sikerült a VU—3D-t megismernem, közzéteszem eddigi tapasztalataimat.

A program használata

A VU—3D lényegesen több billentyűt használ, mint egy átlagos játékprogram, de ennek ellenére a program használata nem bonyolult. Használatát elmagyarázni annál inkább. Hogy ezen könnyítsek, összeállítottam a *táblázatot*. Ebből kiolvasható, hogy egy-egy almenünek milyen funkciói vannak, és ezek milyen billentyűvel aktivizálhatók. Ezért, ha leírás közben eljutunk egy új almenühöz, nem írom le és magyarázom

A VU–3D billentyűkiosztása

Menü / Bill.	1. Modify a figure (Rajzmódosítás)	4. Create a new figure (Új rajz készítés)		6. Display (Kijelzés)	
		Open (alaplaprajz)	Create (testrajz)	Display (kijelzés)	Picture (kép)
"A"					Above (felülről) Sh
"B"					Below (alulról) Sh
"C"			Close (testlezárás)		Centre (középről) Sh Colour (szín vált.)
"D"		Delete (vonal törl.) →			
"E"		End (alaprajz bef.) →			
"F"	Figure (figuraváltás)		Figure (figuraváltás)	Far (képtávolítás)	
"H"					Hidden line (nem látszó él törl.)
"K"					Keep (kép magnóra m.)
"L"		Line (vonalhúzás)			Left (balról) Sh
"M"	Magnify (nagyítás)		Magnify (nagyítás)	Magnify (nagyítás)	
"N"	Next z (z koor. növ.)		Next z (z koor. növ.)	Near (képkitöltés)	
"O"			← Open (kurzor előhív.)		
"P"				Picture (képkimerev.) →	Print (nyomtatóra)
"Q"	Quit (kilépés) ↑		Quit (kilépés) ↑	Quit (kilépés) ↑ ←	Quit (kilépés)
"R"	Reduce (kicsinyítés)		Reduce (kicsinyítés)	Reduce (kicsinyítés)	Right (jobbról) Sh
"S"		Start (kezdő pont ki)			Shade (árnyékolás)
Kurz. bill.	Figuramozgatás	Kurzormozgatás	Figuramozgatás	Képforgatás	
CAPS+ kurz.	Figuram. lassan	Kurzorm. lassan	Figuram. lassan	Képforg. lassan	

el annak minden funkcióját, hanem csak azt említem meg zárójelben, amelyikre éppen szükség van. Például (Magnify). Így fokozatosan, lépésről lépésre magyarázhatom el a program működését.

Aki nagyjából ismeri a VU—3D működését, az a táblázat segítségével mindjárt használni is tudja. A billentyűk használatát segíti az is, hogy az almenüknél megjelenő feliratok kezdőbetűi az adott billentyűkre utalnak (open="O", close="C", delete="D".).

Betöltés

A program betöltése a LOAD "" parancs kiadásával történik. Ekkor betöltődik a loader (EXAMPLE), a BASIC program (example), a címlap *(1. ábra)* és a gépi kódú rész (c).

Betöltés után megjelenik a képernyőn egy pohár és egy kocka térbeli képe, valamint a DISPLAY menü. A képet a kurzorbillentyűkkel körbeforgathatjuk, nagyíthatjuk (Magnify), kicsinyíthetjük (Reduce), közelíthetjük és távolíthatjuk (Near és Far).

A PICTURE üzemmódról később lesz szó, de aki kedvet érez hozzá, a táblázat segítségével már most is kipróbálhatja.

Főmenü

A DISPLAY üzemmódból a Q billentyű segítségével juthatunk el a főmenühöz (Quit). Ez a következő pontokból áll:

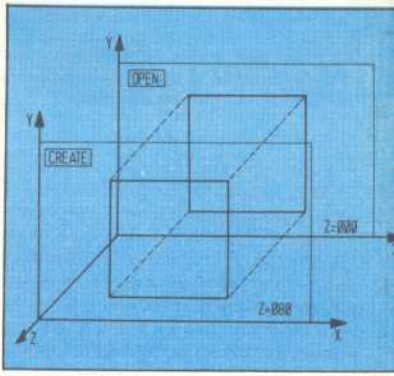
1. Modify a figure = egy kész rajz módosítása
2. Abadon = az elkészült rajz kitörlése (új ábra rajzolása esetén)
3. Load a data file = egy régebbi rajz betöltése magnóról
4. Create a new figure = új rajz készítése
5. Save a data file = a kész rajz kimentése magnóra
6. Display = a rajz térbeli ábrázolása
7. Change colours = új papír-, tinta-, keretszínek beállítása

Próbáljunk meg most egy új rajzot készíteni. Rajzoljunk egy 80 pixel élhosszúságú kockát. Válasszuk a főmenü 4. pontját (Create . . .). Az ENTER-t lenyomva a következő felirat jelenik meg a képernyőn:

File already in existence
Use Abadon to delete it
Press any key to continue

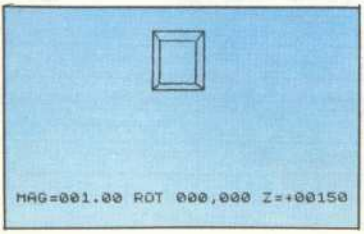
Ez arról tájékoztat bennünket, hogy a pohár rajzát még nem töröltük ki a programból. Egy billentyű lenyomása után visszajutunk a főmenühöz és a 2. (Abadon) pontot választva elvégezhetjük a törlést.

I. ábra. A VU—3D X—Y—Z koordinátarendszere

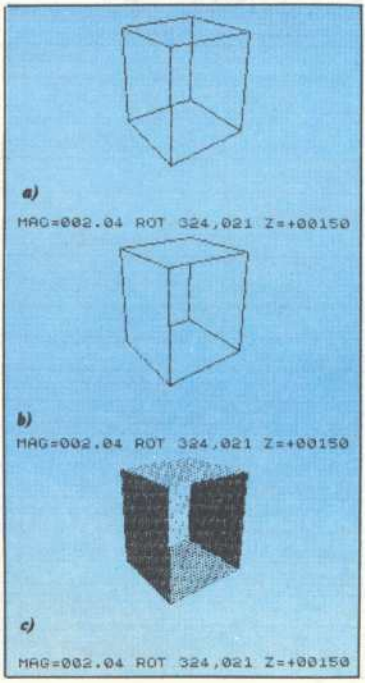


1. ábra

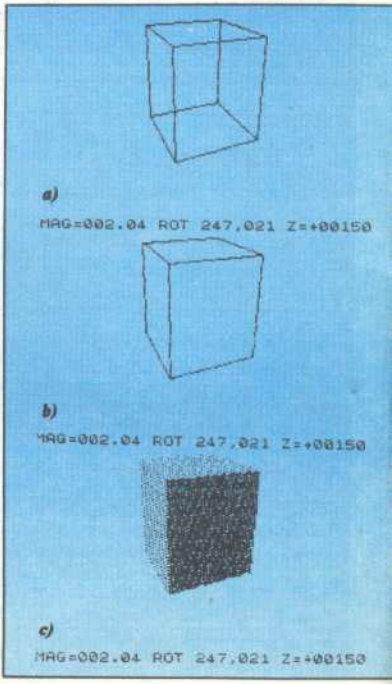
2. ábra

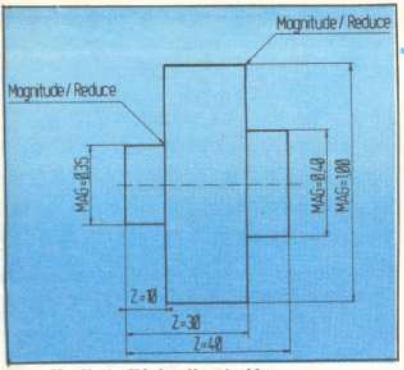


3. ábra



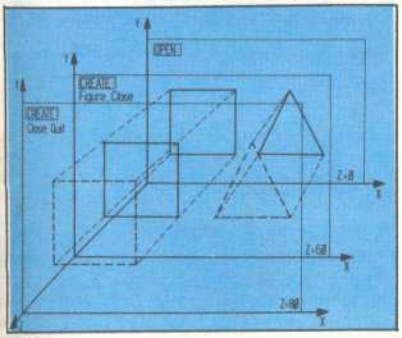
4. ábra





II. ábra. Két hasáb rajzolása

III. ábra. Lépcsős forgástest rajzolása



5. ábra

A törlés megtörténtéről a menü alatt megjelenő FILE ABADONED felirat tájékoztat.

Ebből azt is észrevehetjük, hogy egy test rajzolását nem szakíthatjuk meg csak azért, hogy megnézzük, hogyan mutat a térben. Hiszen ha egyszer kiléptünk a „Create . . ." üzemmódból, oda már csak a rajz kitörlése után juthatunk vissza.

Ezek után már nincs akadálya, hogy elkezdjük a rajzolást.

Kocka rajzolása

A 4. pontot választva a CREATE menühöz és egy síkbeli X—Y koordinátarendszerhez jutunk. A test ábrázolása viszont egy térbeli X—Y—Z koordinátarendszerben történik. Ezt a program írói a következő módon oldották meg.

Az X—Y koordinátarendszerben megrajzoljuk a test alaprajzát. Ekkor még a Z koordináta értéke 0, ami a képernyő jobb alsó sarkában olvasható is. Az alaplap megrajzolása után az „N" (Next z = következő z) billentyűt nyomogatva állíthatjuk be a Z értékét, és ezzel a test vastagságát. *(I. ábra).*

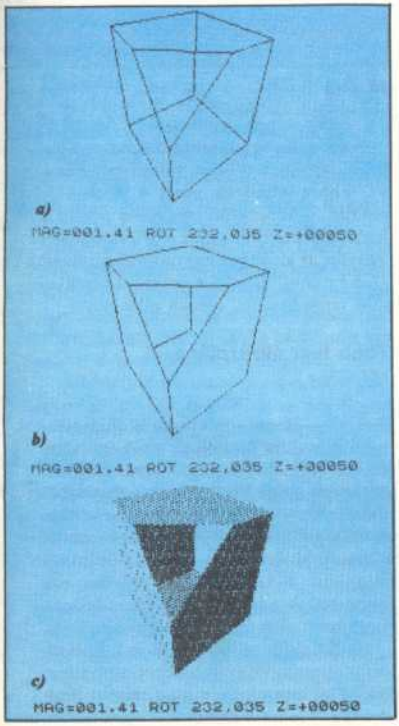
Hívjuk elő a kurzort (Open). Ekkor az OPEN almenühöz jutunk. A kurzor a kurzorbillentyűkkel mozgatható X = 0—240, Y = 0—144 értékek között. A finombeállítás a CAPS SHIFT billentyű és a kurzorbillentyűk egyidejű lenyomásával történik. A mindenkori X—Y érték a képernyő alján olvasható. Állítsuk a kurzort megfelelő helyzetbe, és tegyük ki a kezdőpontot (Start). Rajzoljuk meg egy 80 × 80-as négyzet három oldalát (vonalhúzás = Line, ha elrontottuk, akkor törlés = Delete). Ha a harmadik oldallal is készen vagyunk, az E billentyű lenyomásával fejezzük be a rajzot (End).

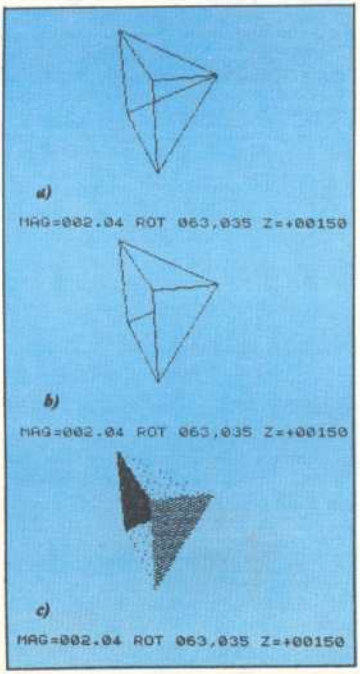
Visszajutunk a CREATE almenühöz, és megjelenik előttünk a teljes négyzet, de nem folyamatos, hanem szaggatott vonalakkal ábrázolva. Vagyis láthatjuk, hogy nem szükséges az egész síkidomot megrajzolnunk. A rajzot a gép — a kezdő- és végpontot értelemszerűen összekötve — befejezi.

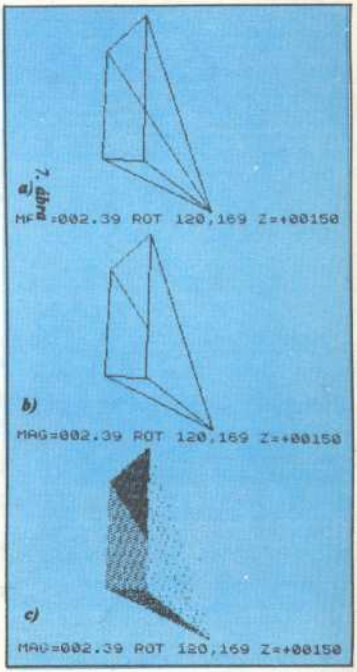
Most állítsuk a Z koordináta értékét 80 pixelre. Fejezzük be a kocka rajzolását (Close). Ekkor megjelenik a DO YOU WANT THE TOP OPEN? Y OR N felirat. Vagyis nyitva legyen-e a kocka felénk néző része? Legyen nyitva. (Y).

Térjünk vissza a főmenühöz (Quit), és nézzük meg a kocka térbeli képét (6. Display), *(2. ábra).*

6. ábra







A kép alján lévő jelölések:
MAG = nagyítás = 1.ØØ
ROT = elforgatás = Y tengely körül = Ø; X tengely körül = Ø
Z = most a képernyőtől való távolság = 15Ø

Nagyítsuk ki a képet (Magnify) és forgassuk el a kurzorgombokkal *(3. ábra)*, majd merevítsük ki (Picture).

Töröljük ki a nem látszó éleket. A H billentyű lenyomása után a fejléc alján megjelenik a villogó HIDDEN LINE felirat, és elvégzi a műveletet. Látható, hogy a kocka egyik lapja tényleg hiányzik *(3.b ábra)*.

Árnyékoljuk be a képet (Shade). A most megjelenő feliratok a fényforrás irányáról érdeklődnek: Above = felülről; Centre = középről; Below = alulról; Left = balról; Centre = középről; Right = jobbról.

Válaszoljunk a megfelelő billentyűk lenyomásával, és a gép elvégzi az árnyékolást *(3.c ábra)*. Árnyékolás közben a fejléc alján villogó SHADING felirat olvasható.

Térjünk vissza a DISPLAY menühöz (Quit). Forgassuk tovább a kockát úgy, hogy nyitott részét elölről ne láthassuk. Készítsünk így is képet róla *(4.a,b,c ábra)*.

Most állítsuk be úgy a kockát, hogy egyik csúcspontja felénk nézzen, és készítsünk így egy metszetet. Közelítsük a kockát a képernyőhöz (Near), és közben kicsinyítsük a képet úgy, hogy az egész kocka elférjen a képernyőn (Reduce), *(5.a,b,c ábra)*. A kép mutatja, hogy a kivágott nyíláson keresztül belátunk a kocka belsejébe.

Mivel rajzunkat alaposan körbenéztük, próbáljunk meg valamit módosítani rajta.

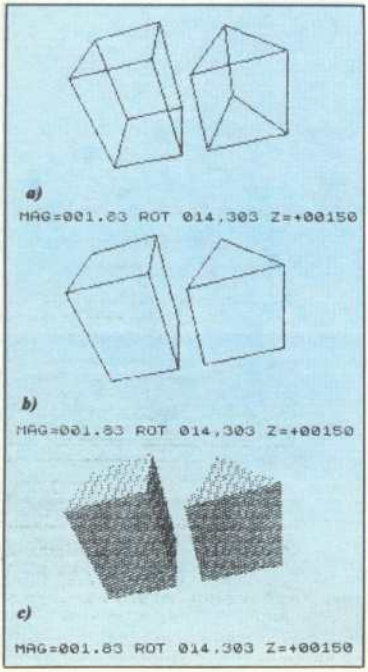
Gúla rajzolása

Készítsünk most kockánkból egy négyzet alapú gúlát. Ezt úgy érhetjük el, ha a kocka egyik lapját ponttá zsugorítjuk.

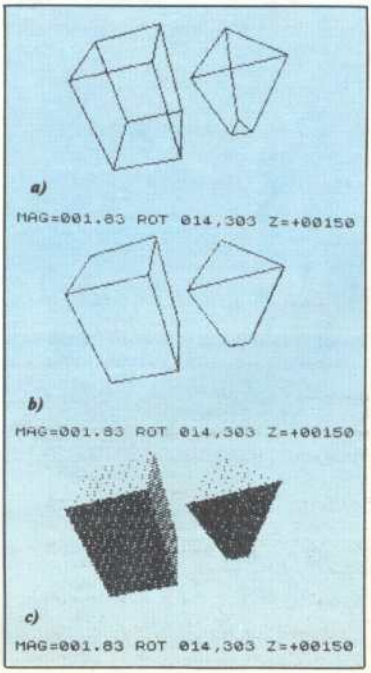
Térjünk vissza a főmenühöz és válasszuk annak első pontját (Modify . . .). Ennél az üzemmódnál előbb a kocka alaplapját látjuk (Z = Ø), majd a Z koordináta növelése (Next z) után a fedőlapját (Z = 8Ø). Mind a két síkidom a kurzorbillentyűkkel mozgatható, nagyítható (Magnify), kicsinyíthető (Reduce). A kép alján olvasható a nagyítás mértéke, a síkidom középpontjának X és Y koordinátája és a Z koordináta értéke.

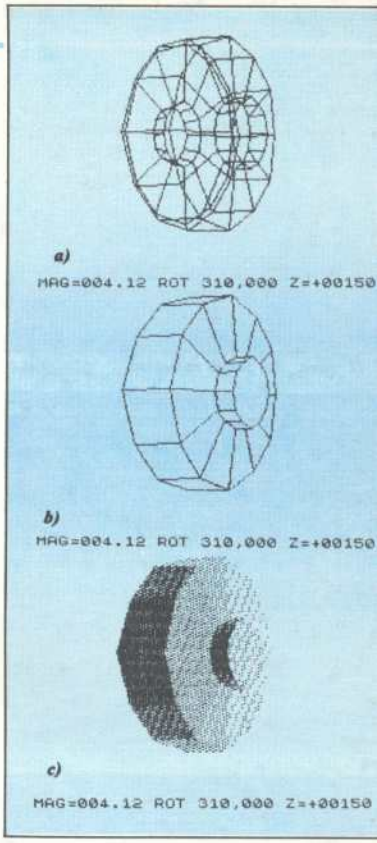
Zsugorítsuk a kocka alaplapját (Z = Ø) egy ponttá (Reduce). Hívjuk elő a fedőlapot (Next z), majd térjünk vissza a főmenühöz (Quit). Ha megnézzük térben ábrázolva rajzunkat, láthatjuk, hogy egy gúlához jutottunk *(6.a,b,c ábra)*.

8. ábra



9. ábra





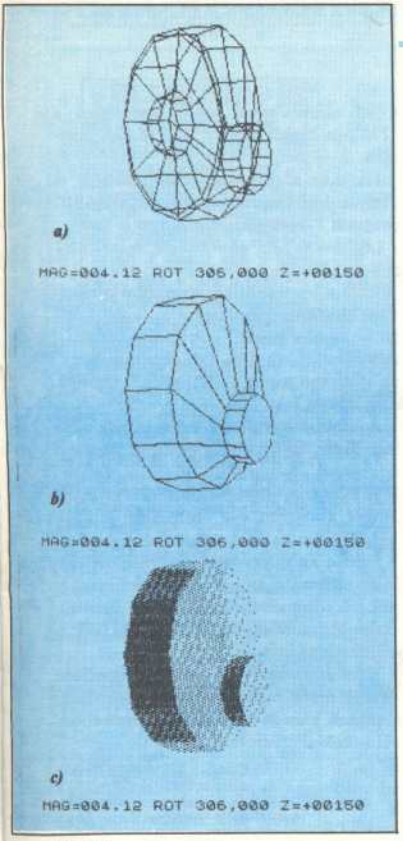
10. ábra

Ferde gúlát úgy készíthetünk, hogy a gúla csúcspontját eltoljuk valamilyen irányba. Ehhez válasszuk ismét a főmenü 1. pontját. A képernyőn megjelenő pontot a kurzorgombokkal toljuk el valamilyen irányba, az alaplapot hagyjuk változatlanul *(7. a,b,c ábra)*. Ezzel a módszerrel készíthetünk ferde hasábot, csonka gúlát stb.

Több test ábrázolása
(II. ábra)

A VU—3D-vel több testet is ábrázolhatunk egyszerre. Rajzoljunk most egy négyzet és egy háromszög alapú hasábot. A négyzet alapú hasáb legyen a magasabb. Először töröljük ki az előző rajzot (Abadon), majd válasszuk a főmenü 4. pontját (Create). Hívjuk elő a kurzort (Open), és rajzoljunk a képernyő bal oldalára egy négyzetet. A négyzet harmadik oldalának meghúzása után fejezzük be a rajzot (End).

Hívjuk elő ismét a kurzort, és a jobb oldalra rajzoljunk egy háromszöget. A második oldal megrajzolása után szintén Enddel zárjuk le a síkidomot.



11. ábra

Azt látjuk, hogy a négyzet szaggatott, a háromszög viszont folyamatos vonalakból áll. Nyomjuk meg az F billentyűt (Figure). Most a négyzet rajza lesz folyamatos és a háromszögé szaggatott. Újabb figuraváltás után ismét a négyzet lesz szaggatott vonalú. CREATE üzemmódban a Close, Magnify, Reduce parancsok és a kurzorgombokkal történő mozgatás mindig a szaggatott vonallal ábrázolt figurára érvényesek.

Növeljük Z értékét 60 pixelre. Most váltsunk figurát úgy, hogy a háromszög legyen a szaggatott rajzú (Figure). Zárjuk le a háromszög alapú hasábot (Close). Növeljük Z értékét 80-ig, és zárjuk le a négyzet alapú hasábot is (Close). Kész is vagyunk, nézzük meg az eredményt *(8.a,b,c ábra)*.

Most készítsünk a háromszög alapú hasábból egy csonka gúlát. Válasszuk a főmenü 1. pontját (Modify). A megjelenő két alaplap közül az előbb már látottakhoz hasonlóan az egyik ismét szaggatott, a másik folyamatos vonalakból áll. Válasszuk ki a háromszöget (Figure) és kicsinyítsük le MAG=Ø.2Ø értékre (Reduce). Növeljük Z értékét 80-ig anélkül, hogy bármi mást változtatnánk. Nézzük meg az eredményt *(9. a,b,c ábra)*.

Forgástestek rajzolása

A VU—3D csak sík felületekkel tud dolgozni, ezért a programmal körök és körívek nem rajzolhatók. Így a kört kénytelenek vagyunk sokszöggel helyettesíteni.

A *10.a,b,c ábrán* látható forgástest egy tizenkét oldalú sokszögből készült. Az ábrán azt érdemes megfigyelni, hogyan változtatható rajzolás közben egy henger átmérője.

Töröljük ki az előző rajzot és rajzoljunk fel a képernyő közepére egy 80 pixel csúcstávolságú, 12 oldalú sokszöget. A 11. oldal megrajzolása után fejezzük be a sokszöget (End), majd kicsinyítsük le MAG=Ø.35 értékre (Reduce). Z értékét növeljük 10 pixelre (Next z). Most valahogy jelezni kell a gépnek, hogy itt egy hengeres test vége következik. Ha egyből felnagyítanánk a sokszöget MAG=1.ØØ-re, akkor egy kúpos felületet kapnánk. Nagyítsuk fel tehát MAG=Ø.4Ø-re, és kicsinyítsük is rögtön vissza MAG=Ø.35-re. Ebből megérti a gép, hogy itt a henger vége, és hogy az alaplap méretén nem változtattunk *(III. ábra)*.

Növeljük a Z-t 1 pixellel és nagyítsuk a sokszöget MAG=1.ØØ-re. Most növeljük Z értékét 3Ø pixelre. Ismét Magnitude/Reduce, majd legyen Z 31 pixel, és csökkentsük a sokszöget MAG=Ø.4Ø értékre. Z értékét 4Ø pixelre növeljük, és végre befejezhetjük a rajzot (Close).

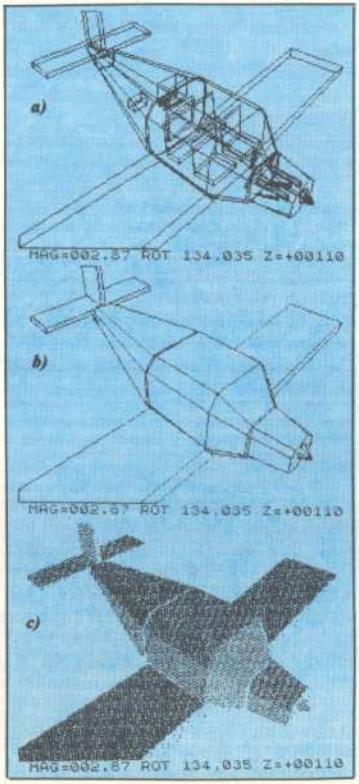
Ezzel a módszerrel készült a program működését bemutató pohár rajza is.

A *11.a,b,c* ábrán látható excenter az előző forgástestből készült úgy, hogy az I. (Modify . . .) opció segítségével az utolsó csapot a kurzorgombokkal excentrikusan eltoltam.

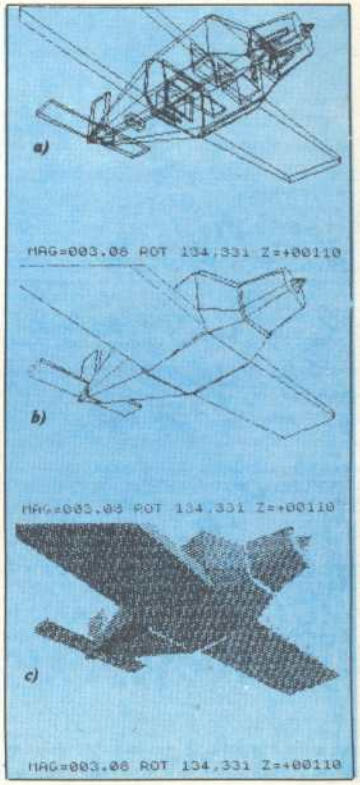
Bonyolultabb testek ábrázolása

A fenti módszereket összekapcsolva bonyolultabb testeket is ábrázolhatunk. Ennek bemutatására készítettem el egy repülőgép röntgenrajzát (12—15. ábra).

12. ábra



13. ábra



Ha csak mint röntgenrajzot nézzük, nem túl bonyolult, de ne felejtsük el, hogy minden irányból megnézhető *(12—13. ábra)*, és metszetek is készíthetők belőle *(14—15. ábra)*. A Spectrumot mindenesetre alaposan megizzasztja. A kép forgatásánál egy-egy képváltás előtt is a nagyítástól függően 4—6 másodpercig számol. Az árnyékolást két, a nem látható élek kitörlését kb. három percig végzi.

Ha a repülőgép törzséről hosszmetszetet akarok készíteni, könnyen megkaphatom a program „Memory Full" üzenetét. Ilyenkor a program nem áll le. A kép is megmarad elforgatott helyzetében, csak éppen MAG = 1.ØØ-es nagyítással, és Z = + 256 pixel távolságra, hogy a számítógép végre egy kis levegőhöz juthasson.

Egy tanács: Mint már említettem, egy test rajzolását a programban nem lehet megszakítani, hiszen ha kilépünk a 4. opcióból, oda már képtörlés nélkül nem juthatunk vissza. Ezért egy bonyolultabb testet jó papíron megtervezni, mielőtt leülnénk a géphez.

A rajz kimentése

Elkészült rajzunkat több módon is megörökíthetjük. PICTURE üzemmódban a kimerevített kép screenként magnószalagra menthető (Keep). Ilyenkor a gép felteszi a „Title (név)" kérdést. Rajzunknak egy nevet adva, nyomjuk meg az ENTER-t. Megjelenik a „Start tape . . ." üzenet, és a kép a szalagra kerül a pillanatnyi nagyítás, elforgatás és Z távolság értékével. Az így kapott képernyőkép az üres gépbe is betölthető (LOAD""SCREEN$), felhasználható saját program címlapjaként stb.

Szintén PICTURE üzemmódban a kimerevített képet ki is nyomtathatjuk (Print). A mozgatható képet a főmenü 5. (Save a data file) pontja segítségével menthetjük magnóra. Forgassuk a képet a nekünk legjobban tetsző helyzetbe, mert így kerül a szalagra. Ezután válasszuk az 5. pontot és végezzük el a kimentést.

Az így kitöltött rajz csak a VU—3D programba tölthető vissza a főmenü 2. pontja segítségével. Magnótöltési hiba esetén a program hibaüzenettel leáll. Ilyenkor GOTO 2ØØØ paranccsal a főmenühöz jutunk vissza, RUN paranccsal a DISPLAY üzemmódhoz.

A színek beállítása

A program színeit kétféle módon állíthatjuk be:

1. A főmenü 7. (Change colours) pontja segítségével. Ilyenkor képernyőtörléssel állnak be az új színek.
2. A PICTURE üzemmód Colour utasításával. Ilyenkor az új színek képernyőtörlés nélkül állíthatók be, és szintén az egész programra érvényesek.

A feltett kérdések:
Border colour = keretszín
Banner paper = fejlécpapír
Banner ink = fejlécceruza
Screen ink = képernyőceruza
Screen paper = képernyőpapír

A színek beállítására az ismert színkódokat használjuk.

Egy-két ötlet

Ha készítettünk egy olyan rajzot, amely jobban tetszik, mint a programban lévő pohár és kocka (pl. 12—15. ábra), elérhetjük, hogy a VU—3D betöltése után egyből a mi rajzunk jelenjen meg a képernyőn.

Töltsük be a gépbe a VU—3D-t, majd a saját rajzunkat (2. Load a data file). A főmenünél lépjünk ki az idézőjelből és STOP-pal állítsuk meg a programot. Ezután vegyünk elő egy kazettát, és töltsük ki az egész gépi kódú részt: SAVE „c" CODE 31 5Ø1, 32 183. Így lesz egy VU—3D gépi kódú részünk, ami ragyogóan működik, és a mi rajzunk van benne, méghozzá a 32 812-es címtől kezdve, 12 33Ø bájt hosszan.

Ha még a program kezdőszíneit is át akarjuk írni, ezt is megtehetjük a BASIC program 7. sorában:

7 LET b = 5: LET pb = 1: LET ib = 7: LET ps = Ø: LET is = 7

Hála a beszédes változóneveknek, könnyen megfejthetjük, hogy a
pb = banner paper
ib = banner ink
ps = screen paper
is = screen ink

A színkódok ismertek, rajta vannak a billentyűkön. Javítsuk még át az 512Ø-as sort is:

512Ø LOAD „c" CODE: BORDER x: RUN

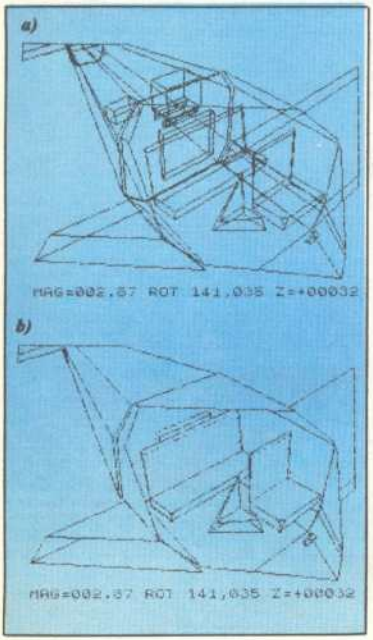
ahol x szintén egy tetszőleges színkód Ø-tól 7-ig.

Ezután másolóprogram segítségével szalagra tölthetjük az egész programot a következő sorrendben: EXAMPLE (eredeti), example (módosított), címlap (eredeti), c (módosított).

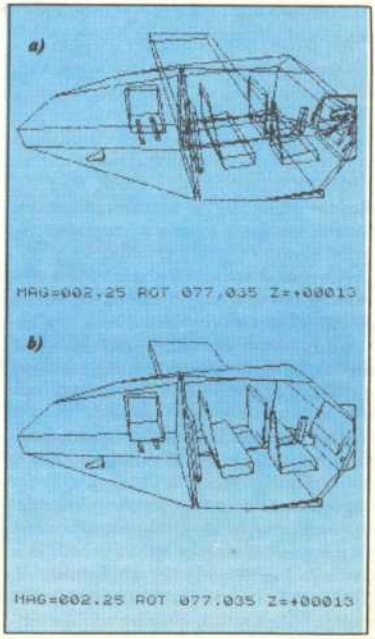
Ezzel leírásom végére értem. Remélem, sikerült újabb ötletet nyújtani a Spectrumhívőknek ahhoz, hogy számítógépüket minél sokoldalúbban használhassák.

LITAUSZKY GYÖRGY

14. ábra



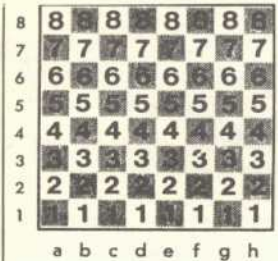
15. ábra



BITEK ÉS FIGURÁK

Állásértékelés VIII.

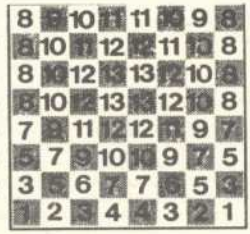
Támadó és védekező stratégiájú programok



1. a) ábra. Támadó súlyozás világossal

A már említettek alapján tudjuk, hogy egy támadó stílusban játszó sakkprogramnak nagyobb esélye van a nyerésre, mint egy védekező stratégiájúnak. Ugyanis a jelenlegi programok gyengéje a középjáték és a végjáték közötti átmenet. Ezt elkerülhetjük éles támadó játékkal, melyben felboritjuk a pozíciós egyensúlyt, és ezzel a parti végső kimenetele gyakran már a középjátékban eldől.
Ezt a támadó játékot a megfelelő időpillanatban kell elkezdeni. Ennek a pillanatnak a meghatározása nehéz feladat, mivel nagyon sok tényezőt kell figyelembe venni. Külön kell kezelni a zárt és a nyílt állásokat, mindkettőnél más-más szempontokat kell nagyobb súllyal számba venni. Zárt állásnál, ahol a centrumban a gyalogok rögzítik egymást, a szárnyi támadásokat kell előkészíteni, amelyek általában gyalogrohammal kezdődnek. A gyalog-előrenyomulás a tisztek támogatásával a leghatékonyabb. Majd miután lecserélődtek a gyalogok, akkor veszik fel egymással a küzdelmet a tisztek a nyílt vonalakon.
Nyílt állásoknál a gyalogroham veszélyes, mert lassú, és akadályt képezve elzárja a nyílt vonalakon mozgó gyors tisztek útját. Ezzel a támadás lelassul, és így lehetőséget ad az ellenfélnek, hogy átvegye a kezdeményezést.
Mozgó, labilis centrum esetében a legnehezebb meghatározni a támadás stratégiáját, mivel egy ilyen nem tisztázott pozíció átmenet az

1. b) ábra. Támadó súlyozás sötéttel



2. a) ábra. A támadó és a centrum. Súlyozás világossal

előbb említett egyik állástípusba, ahol más-más módszerrel kell támadni. Ezért a jelenlegi sakkprogramok az ilyen hadállásokat ütésekkel vagy a gyalogok rögzítésével igyekeznek megszüntetni. Ez nem helyes, mert ezzel sokszor hátrányosabb pozíciót vállalnak magukra.
Most viszont nézzük a támadó játék motiválásának legegyszerűbb esetét! A centrumellenőrzés mértékének megállapításánál már megismerkedtünk a sakktábla súlyozásának egyes módszereivel. Jelenleg is egy hasonlót fogunk használni, csak azzal a különbséggel, hogy a súlyozó értékek nem a centrum felé fognak növekedni, hanem a támadás célpontja felé. Egy ilyen célpont lehet például az ellenfél térfele egyenletes eloszlással. Ilyen súlyozást látunk az *1. a,b ábrán.*
Az ilyen egyenletes eloszlású táblának hátránya, hogy a figurákat az ellenfél tábora felé irányítja, különösebb célpont nélkül. Ezáltal a különböző szárnyakon lévő figurák az ellenfél térfele felé igyekeznek anélkül, hogy bármiféle összehangoltság lenne közöttük. Így csak azt érjük el, hogy a saját térfelén lévő mezőket túlságosan meggyengitjük, és ezzel lehetőséget adunk az ellenfélnek, hogy ezeken a gyenge mezőkön megvesse a lábát és behatoljon állásunkba.
Tehát ha mégis ezt a módszert alkalmazzuk, akkor még más tényezőket is figyelembe kell venni, ami

2. b) ábra. A támadó és centrum. Súlyozás sötéttel

biztosítja a figurák együttműködését: jutalompontot adhatunk például akkor, ha az ellenfél térfelén lévő egyik mezőt két vagy több különböző bábuval támadjuk. Hasonlóan akkor is, ha az egyes figurák által megtámadott mezők „közel" mennyiségi megállapítása a programbelövés feladata.) Ilyen módszerekkel megakadályozhatjuk haderőnk széthúzását, szétesését. Igaz, több olyan kombináció is előfordul a sakkban, aminek lényege, hogy egyetlen figurával megtámadjuk az ellenfél két figuráját, és ezáltal anyagi értéket nyerünk, vagy matthelyzetet hozunk létre. Ez azonban változatszerű számítást igényel, ezért a program ezt nem hagyja ki, annak ellenére, hogy tisztjei kis időre eltávolodtak egymástól, és így harmóniájuk felbomlott.
Precízebb táblasúlyozást kapunk, ha a centrumtáblát összevonjuk ezzel a támadó súlyozású táblával, vagyis a kettőt összeadjuk *(2. a,b ábra).* Ezzel időt takarítunk meg, mert nem kell külön-külön számítani az értékeket.
Ugyancsak időt takaríthatunk meg, ha ezt még a királytámadási táblával is összevonjuk *(3. ábra),* csak ennek hátránya, hogy minden királylépés esetén újra kell a súlyozó értékeket számítani.
A tapasztalat mégis azt mutatja, hogy megéri ez a megoldás, mert a király egy játszma folyamán kevesebb lép. Az algoritmusba, mely a lépésvariációkat vizsgálja, viszont beépíthetjük, hogy csak akkor módositson ezen a súlyozáson, ha a király az eredeti helyétől legalább két lépést megtett. Ez nagyon ritka eset, mert egymás után kétszeri királylépésre általában csak a végjátékban kerül sor, ahol viszont a királytámadási súlyozás már felesleges. Nincs anyag a mattadáshoz; ilyenkor a gyalogbevitel a cél. Erre viszont a sima támadótáblát célszerű használni, mert ezzel a programot a gyalogok előretolására késztetjük, így gyorsítjuk a gyalogbevitelt.
Az előbbiek alapján el tudjuk képzelni, hogy ezen az alapon védekező stratégiájú programot is lehet készíteni. Ebben az esetben viszont magasabb értékkel a saját térfélen lévő mezőket látjuk el. Ezáltal a program kerülni fogja a gyalogokkal való előrenyomulást, és a tisztjeit is a saját alapsora közelében helyezi el.
A megfelelő táblasúlyozás kikísérletezése sok időbe kerül, míg megtaláljuk a leghatásosabbat.
A statikus állásértékelés általában egy szimmetrikus függvény. Ez azt jelenti, hogy amilyen szempontot az egyik színnel számításba vettünk, azt a másik színnel is számításba kell vennünk, csak ellentétes előjellel. Tehát egy értékelésnél mindkét színnel figyelembe kell

3. ábra. Támadó és királytámadási súlyozás összevonása

venni a centrumértéket ugyanúgy, mint a támadóértéket. A világos és sötét pontszámának különbsége adja magának az állásnak az ilyen komponensekből álló pozíciós értéket.
Ezt a szimmetriát viszont megbonthatjuk a támadótábla súlyozásával. Egy játszma alatt mind a két színnel külön-külön, másfajta súlyozást is használhatunk. Például a program figyelheti az ellenfél játékát, és ennek mértékében súlyozza a tábláját. Vagyis, figyelemmel kíséri az egész játszmát, és az előző lépések alapján kielemzi az ellenfél játékstílusát, amely lehet védekező, támadó, közömbös vagy várakozó. Itt a közömbös vagy várakozó játékstíluson azt értjük, hogy a játékos mindenféle konkrét terv nélkül lépked figuráival, és az ellenfél akciójára vár.

KOVÁCS P. ATTILA

GÉPI ELLENFELEINK

Chess Challenger

A Chess Challenger Sensory 9

Legutóbbi cikkeinkben az NSZK-beli Hegener+Glaser népszerű Mephistóit ismertettük, amelyek jelenleg a világ élvonalát képviselik. Utolérték és el is hagyták az USA-beli Fidelity készítményeit, amelyek éveken át világelsők voltak. Fidelity azonban ma is a legerősebb versenytárs.

Történelmi érdem

Anélkül, hogy a számítógépes sakknak vagy akár csak a mikroszámítógépek sakktudásának fejlődését ismertetni kívánnánk, szólnunk kell a legelső sakkmikrók megjelenéséről. Hiszen azóta bő tíz esztendő telt el, és az első készülék neve ma is közszájon forog.

E sorok írójától sokan érdeklődnek bizonyos típusú sakkszámítógépek képességei, ára felől, mert el szeretnének adni egy régi készüléket, vagy vásárolni akarnak egy újat. De ha a kérdező csak azt mondja, hogy Chess Challenger típusú készüléke van, ne csodálkozzon a kitérő válaszon. Ilyenkor ugyanis csak azt mondhatjuk: ez körülbelül úgy hangzik, mintha arra a kérdésre, hogy milyen autója van, azt mondaná: Volkswagen. A Chess Challengereknek is tucatnyi fajtája került ugyanis az évek során forgalomba, és az elsőtől az utolsóig akkora volt a változás, mint mondjuk a legöregebb bogárhátú Volkswagentől a Golfig.

Az első gyár, amely mikroprocesszorral működő sakkszámítógéppel megjelent a piacon, a Miamiban székelő *Fidelity Electronics* (ma Fidelity International) volt. 1976-ot írtak: a készüléket Ronald C. Nelson építette. A számítógépes sakk történelmi fordulója volt ez! Csakhamar megjelentek a továbbfejlesztett típusok: a *Chess Challenger 3* és *10*, majd a *7*.

Már itt felhívjuk az olvasó figyelmét, hogy a típust kísérő számokból semmire sem lehet következtetni. Ezeket teljesen ötletszerűen bigygyesztették a név mögé, és amint a továbbiakból is kiderül, a magasabb szám nem jelent feljettebb vagy akár újabb típust. Az eddig felsoroltak — a 7 is, amelyből még néhol megjelenik egy-egy példány —, mind elavultak. Ugyanezt mondhatjuk a szám nélküli, Voice vagy Sensory melléknévvel ellátott készülékekre, valamint a Decoratorra, melyek itt-ott még szabálytalan lépéseket is tesznek.

A Chess Challenger Champion Sensory, nyomtatóval és Gedeon Béla fehér dobozban elrejtett gyorsítójával

A technikai fejlődést azonban tükrözi, hogy a Voice hangot ad, bemondja a megtett lépést, néha megjegyzést is fűz hozzá, tréfásat, szórakoztatót; a Sensory pedig, amint nevéből is következik, az első szenzoros típusú Chess Challenger volt. Ehhez nem kellett külön sakktáblát használni a lépések megtételére. A számítógép maga, akárcsak a régi Mephistók, nyomógombokkal működik, és kijelzője mutatja a megtett lépéseket. A Decorator nem más, mint a régi Voice, csak díszesebb kivitelben: fából készült, nem műanyagból, mint a többi, olcsóbb készülék.

Ha az olvasónak ezek közül kínálnak egyet megvételre, azt ajánljuk, ne foglalkozzék vele. Az első Fidelity gyártmányú sakkszámítógép, amely ma is versenyképes, és jó sakkozó számára is partner, a *Chess Challenger Champion Sensory* nevet viseli.

Az első sakkvilágbajnok

1980-ban, Londonban rendezték a mikroszámítógépek első sakkvilágbajnokságát, amelyet a győztelem után Champion melléknévvel ellátott Chess Challenger nyert meg. Ez azonban nem továbbfejlesztett, hanem merőben új készülék volt. Megszületésének előzménye: a Fidelity alkalmazásába lépett Kathe és Dan *Spracklen*, akik a Sargon programokkal elért kiemelkedő sikereikkel a mikroszámítógépes sakknak ekkor már koronázatlan királyai voltak. A Sargonokkal itt nem kívánunk foglalkozni, mert ezek — a Fidelity részére továbbfejlesztett programokkal párhuzamosan — továbbra is a piacon maradtak, és megjelentek újabb verzióik is. A Sargonokra tehát visszatérünk.

A Spracklen házaspár teljesen új programot készített a Champion számára. A megnyitási könyvtárat, a lépésgenerátort és az értékelőfüggvényt — tehát lényegében a programnak minden részletét — újrarírták. Kathe Spracklennek saját, annak idején hozzánk intézett soraiból idézünk: „A program a korábbinál gyorsabb, taktikai készsége jobb, könnyebben továbbfejleszthető újabb sakkbeli kritériumokkal, egyszerűbb a hadállások felállítása." A hardver sebességét is a korábbi 1-1,5-ről 2 MHz-re emelték.

A piacon a CC Champion Sensory számított hosszú időn át a legjobb sakk-mikroszámítógépnek. A következő, Travemündében rendezett világbajnokság „kereskedelmi" csoportjában a Scisys cég Mark V-je megelőzte ugyan, de ennek hardverproblémái miatt változatlanul a Champion volt a világszerte legelterjedtebb sakkszámítógép. Technikai szempontból is megelőzte vetélytársait, hiszen amint legutóbb írtuk — a Mephisto készülékek, s a NOVAG számítógépei is később tértek át a szenzoros módszerre. Igen könnyen kezelhető; az indulási és érkezési mező könnyed megnyomásával kell a lépéseket megtenni, s ezeket — csakúgy, mint a gép válaszlépéseit — a mezők bal alsó sarkában levő dióda kigyulladása jelzi. A bábszimbólumok megnyomása útján lehet hadállásokat is felállítani. Lépéseket vissza lehet venni (legfeljebb 19-et), a sakkóra akár a felhasznált, akár az időkontrollig még rendelkezésre álló időt jelzi. Számol azalatt is, amíg az ellenfélen a lépés sora. Kilenc játékfokozata van, ami különböző gondolkodási időket jelent a lépésenkénti 10 másodperctől a 3 perces versenyfokozaton át a végtelenig, vagyis az időhatár nélküli elemzésig, és van mattkereső játékmódja, amely feladványok megfejtésére szolgál. Vagyis rendelkezik mindazokkal a legfontosabb tulajdonságokkal, amelyeket egy korszerű számítógéptől elvárunk. A tábla és a készlet fából készült. Eredeti ára 1000 DM körül volt. Ma már nem árusítják, alkalmi vételben 3-400 DM körüli árat kérnek érte, amit feltétlenül megér.

Mint érdekességet említjük meg, hogy Gedeon Béla, a Mikroelektronikai Vállalat gyöngyösi gyáregységének osztályvezető mérnöke készített a készülékhez egy kiegészítő, gyorsító berendezést, amely a mélyebb számítást segíti elő. Készülékével — a Fidelity gyár hozzájárulásával — részt vett az 1983-ban Budapesten rendezett 3. mikroszámítógép világbajnokságon, és 4 pontjával az élmezőnyben végzett, pedig valamennyi résztvevő közül a legrégebbi típust képviselte.

Sensory 9

1983-ban jelent meg az üzletekben a Chess Challenger Sensory 9 típusú sakkszámítógép, amelyet az 1983. évi vb-n ismerhettünk meg. A számos nagyobb teljesítményű kísérleti készülék között 3 pontot gyűjtött, ezzel elnyerte a legjobb, kereskedelmi forgalomban lévő készülék részére kitűzött különdíjat. (Gedeon Béla készüléke kísérletinek volt minősítve.) Azóta számos teszt, versenyeredmény és egyéb tapasztalat bizonyította, hogy ez a készülék ugrásszerű fejlődést jelentett az ember gépi ellenfeleinek sorában. Már megjelenésekor közölte a Spracklen házaspár, hogy kísérleti párosmérkőzésen a Sensory 9 a Campionnál 15:5 arányban bizonyult jobbnak. Egyszerű műanyag kivitele és a fejlettebb program által lehetővé vált technikai egyszerűsítésének következtében a Championnál lényegesen olcsóbban, 4-500 DM körüli áron került forgalomba. Meg kell említenünk, hogy az amerikai készülékek NSZK-beli ára — amely mellett a könnyebb összehasonlítás kedvéért, s mivel honfitársaink a leggyakrabban ott vásárolnak, megmaradunk —, a dollár árfolyamának ingadozása következtében gyakran változott.

A Sensory 9 16 k ROM és 2 k RAM méretű tárolóval rendelkezik, szemben a Champion 20 k ROM és 3 k RAM-jával. 1,5 MHz-en fut; a Champion, amint említettük, 2 MHz-en. Nem ad hangot, s — ami nem előny —, nem jelzi az időt. Belső órája természetesen van, hiszen szintén kilenc fokozatra állítható. Új képessége, hogy jelzi a döntetlent minden olyan esetben, amikor a sakkszabályok azt előírják. Tehát felismeri a patt-ot, a háromszori hadállás-ismétlést, és ismeri az 50 lépéses szabályt is. Egyenlő állás esetén döntetlent ajánl, illetve ilyen ajánlat fölött dönt. Reménytelen állásban a játszmát feladja. Mindezeket a diódák kigyulladása jelzi, megfelelő alakzatokban. Akkor is jelez, ha hét lépésen belül — bármilyen fél részére — védhetetlen mattot lát. Mindezek mellett jelentősen nőtt a korábbi típusokhoz képest a készülék játéktudása, főként taktikai téren. Mintegy 3000 hadállást tartalmaz megnyitástára, s korlátozottan moduláris szerkezete lehetővé teszi kiegészítő — például megnyitási — modulok beiktatását.

Újonnan ezt a készüléket sem árusítják már. De feltétlenül megfelel egy közepes sakkozó igényeinek, és a 2-300 DM körüli alkalmi vételt érdemes meggondolni. A Sensory 9-et leginkább a gyár saját újabb típusai szorították ki a piacról, elsősorban a hatalmas karriert befutott Elite-sorozat, amelynek ismertetésére legközelebb kerül sor.

LINDNER LÁSZLÓ

Sokszor beszélünk arról, hogy az „informatikai forradalom" mennyire megkönnyíti majd munkánkat a nem is olyan távoli jövőben. De erre fel kell készülni, mert nincs annál veszélyesebb, mint amikor valaki olyan eszközt kap a kezébe, amelynek használatára még éretlen.

A kiszolgáltatott ember

A tokiói utca képe megváltozott a május eleji csúcstalálkozó közeledtével. A szokásosnál több rendőr teljesített szolgálatot. A rendőrség felszerelése korszerűsödött, elsősorban a számítástechnika lehetőségeinek felhasználásával, ami a mikroelektronika hazájában nem meglepő. A japán hatóságok több mint harmincezer, szélsőséges nézeteket valló potenciális terrorista adatait tartják nyilván, akiknek jelenléte a csúcstalálkozó biztonsága miatt nem volt kívánatos. Tokióban is mindennapossá vált a Svédországban megszokott kép: a személyi igazolvány adatait vagy egyéb ismertető jegyeket bebillentyűzik a rendőrségi autó terminálján vagy a rendőrök rádiókészülékének billentyűzetén, és máris az intézkedő közegek rendelkezésére áll mindaz, amit a gyanúsnak tűnő emberről a hatóság adatbankjaiban nyilvántart.

A technika azonban ennél is érdekesebb ellenőrzési rendszerek kidolgozására is lehetőséget nyújt. Angliában és az NSZK-ban sikeres kísérletek folynak olyan távékamerás komplex rendszerek kifejlesztésére, amelyek képesek felismerni az előttük elhaladó gépjárművek rendszámait és összevetni a körözött, valamint a megfigyelt járművek jegyzékével. Egyezés esetén riasztják a mozgó őrszemeket.

Az elektronizálás nemcsak az üzemek védelmét könnyíti meg a jogtalan behatolástól elektronikus kártyával, hanggal, ujjlenyomatta vagy mondjuk a szemfenék érmintázatát felhasználó azonosító berendezésekkel. Egyes országok elektronikusan olvasható útlevelet készítenek vagy éppen géppel feldolgozható vízumot ütnek be a különben hagyományos útlevélbe. Így amikor a bevándorlási hatóság útlevélvizsgálója látszólag hanyagul néhány pillanatra szöveggel lefelé kinyitva a pultra teszi az útlevelet, az azonosítás automatikusan végbemegy, minimálisra csökkentve a tévedés esélyét, és egyúttal megnehezítve az útlevél-hamisítást is. Nem elég ugyanis tökéletes, mindenben megfelelő kópiát készíteni, a hamis adatokat valamilyen módon be kell vinni a számítógépes rendszerbe is.

Amint az állampolgár egy informatikai társadalomban kapcsolatba kerül az állammal vagy vállalatokkal, kitörölhetetlen azonosítót kap, amely egész életét végigkíséri, és legbensőbb magánéletének titkai is feltárulnak, ha valamely hatóságnak szüksége van rá. Ehhez senkinek nem kell külön hatékony apparátust üzemeltetnie, csak rendszereznie kell a szétszórtan meglévő részinformációkat.

Pár esztendeje azt mondta valaki az egyik számítástechnikai kongresszuson: „A hitelkártya és a bankcsekk a legtökéletesebb eszköz, amely nyomon követheti az egyén mozgását a világban."

Mr. Smiss évek óta az American Express hitelkártyájával rendelkezik. Különmunkáiból származó jövedelme és fizetése bankszámlájára érkezik, ahonnan rendszeresen automatikusan átutalják hitelkártyájának fedezetét. Mr. Smiss számára mindez nagyon kényelmes. Nem kell magánál készpénzt hordania, fizetéseit a kártya számának telefonba történő bemondásával vagy a kártya felmutatásával egyszerűen végzi. A kártya használatát különleges kódjai és a legújabbakba már beépített mikroprocesszor teszi egyszerűvé és biztonságossá. Smiss úr élvezi a kényelmet. És ahol megfordul, nyomot hagy maga után, amely belekerül a számítógépek emlékező egységeibe. Egy idő után a hitelkártya-intézet elkészíti mozgásprofilját, és ha például a különben városából alig kimozduló Smiss kártyáját egyik nap Párizsban, másnap mondjuk a Bahama-szigeteken használják fel, a számítógépes rendszer jelzi az elfogadónak, hogy valami nincs rendjén, igazoltatni kell a kártya felmutatóját.

Smiss úr közben észre sem veszi, hogy bevételeit ellenőrizheti az adóhivatal fürkész szeme, vásárlásaiból megállapíthatják jellemét, családi körülményeit, de még egészségi állapotát is. A bankszámla forgalma nemcsak arról ad elégazítást a hatóságoknak, hogy milyen jövedelmétől kopaszthatják meg az állampolgárt, hanem a bankoknak támpontot adhat arra, mennyire fegyelmezett fizető, mennyire hitelképes az ügyfelük.

A telefonon vagy csomagküldő áruházakból történő rendelések szintén nyomot hagynak a számítógépek memóriájában. Értő kezekben ezek a különben jelentéktelen dolgok meghatározzák, milyen címlistára kerül fel, konzervatív vagy liberális beállítottságú-e (mert akkor ez vagy amaz a párt szólítja fel reklámlevélben, hogy szavazzon rá), vagy egyáltalán érdemes-e a biztosítóintézeteknek betegbiztosítási vagy életbiztosítási ajánlattal megkeresni őt. Ilyen információk a kórházak és a betegbiztosítók adatbankjából is beszerezhetők.

Mr. Smissről halmozódnak az adatok. Anélkül, hogy valaki személy szerint foglalkozna vele, minden lépését szemmel tartják. Munkahelyének számítógéprendszere nyilvántartja ellankadásait, késéseit, pihenőidejét. Esetleg automatikusan adja a fegyelmit és az elbocsátást. De mindez, amíg az információk nem kerülnek egy helyre, nem nagy baj. Az adatok ugyanis csak összességükben képeznek olyan titkot, amely a magánélethez való jogot sérti. Ide azonban már nem kell más, mint hogy valahol valaki kiadja a parancsot, hogy a még- (szerencsére) külön működő adatbázisokat össze kell kapcsolni ezen a területen is. Így az állampolgár ideálisan ellenőrzötté és irányítottá válhat. Remélhetően elég érett lesz az emberiség ahhoz, hogy ezt megakadályozza.

Az információkkal való kereskedés a fejlett tőkés országokban jelentős üzletággá vált. Cégek adnak és vesznek különböző szempontok szerint összeállított címlistákat. A magánélet helyenként annyira összeszűkült, hogy már nem magánügy: ki, mit, mennyiért és hol vásárolt, vagy milyen az egészségi állapota.

A beszédfelismerő rendszerek, az automata telefon- és telekommunikációs központok fejlődése is kétoldalú. Egyrészt megkönnyítik az ember és a technika, valamint az ember-ember kommunikációt, de egyúttal lehetőséget nyújtanak egy ország teljes hírközlésének ellenőrzésére is. Már korábban is figyelni tudták egyes gyanús személyek telefonbeszélgetését, de ez nem volt általános. Később megteremtődött annak a lehetősége, hogy olyan ellenőrző berendezéseket készítsenek a mikroelektronika eszközeivel, amelyek egy-egy „hívószó" alapján rögzítik azt a beszélgetést, amelyben az a szó elhangzott, és felhívják rá az illetékesek figyelmét, hogy alaposan, emberi munkával ellenőrizzék le a hívót.

A távközlésbe való technológiai beavatkozás ezen túlmegy. Mi szerencsére ennek csak egy olyan primitív formáját ismerjük, amikor a vállalat ügyintézőinek telefonjairól a „takarékoskodás" örve alatt kiköttetik a távhívás lehetőségét. De mit szóljon az az angol munkavállaló, akinek minden munkahelyi telefonját dokumentálja egy újonnan kifejlesztett elektronikus központ, és egyben megakadályozza a gyakran hívott magántelefonszámok felhívását is. Mint a gyártó cég önelégülten hangoztatta: rendszerük tízezer számot képes nyilvántartani.

A pénz is „szagot kapott" az elektronika fejlődésével. Korábban, ha útját ellenőrizni akarták, ezt a számok és sorozatok hosszadalmas kézi listázásával, majd e „tengeri kígyó" számjegyzékek figyelésével tehették meg. Így érthető, hogy egy-egy átadott váltságdíj vagy bankrablásból származó pénz felszívódhatott. Rövidesen ez is múlttá válik. Egy-két éven belül a dollár bankjegyek magasabb címleteinek felső részén fémes szürke sáv fog megjelenni: a pénzre géppel olvasható formában holografikusan rögzített sorozat- és sorszám. Ezután már csak elektronika kérdése, hogy akár minden bankjegy forgalma külön-külön is követhetővé váljon.

A pénzátutalási rendszereknek kényelmünket, anyagi biztonságunkat, a kommunikációs rendszereknek egymás jobb megértését kell(ene) szolgálniuk. Felnőttünk egy új informatikai kor küszöbére, de a felnőttség önmagában nem jelenti azt, hogy tudunk is bánni magával az eszközzel. Ezt meg kell tanulnunk úgy, ahogy a vadász megtanul bánni a puskával. Ahhoz azonban már kellő etikai érzék és morális felkészültség kell, hogy az új eszközt pillanatnyi haragunkban, félelmünkben, bizonytalanságunkban ne forditsuk embertársaink ellen.

Ha a számítógéprendszerek nem hibáink feltétlen dokumentálását — mint némelyik rugalmas munkaidőt nyilvántartó rendszer teszi —, hanem munkánk segítését, az összefüggések megkeresését, a mechanikus munka alóli való megszabadulást jelentik, akkor képesek leszünk valóban emberhez méltóan belépni abba az elektronikus korba, amely nemsokára nálunk is megkezdődik.

KIS JÁNOS

A személyi számítógépes helyi hálózatok gyártói egyre újabb rendszereket jelentenek be, és növekszik a régebbi rendszerek modernizált vagy javított változatainak száma is. A jelenlegi irányzat egyértelműen az IBM vonal erősödése, ezért sorozatunk e befejező része is elsősorban az IBM kompatíbilis helyi hálózatokkal foglalkozik. (A sorozat az 1986. szeptemberi számban indult. — A szerk.)

Néhány „nem igazi" helyi hálózatról is szó lesz, amelyek bizonyos mértékig eltérnek az előző részekben ismertetett klasszikus helyi hálózati megoldásoktól, de ezeket kedvező áruk és szolgáltatásaik miatt ma is sok helyen alkalmazzák.

A Net One hálózat

A Net One hálózat az Ungermann-Bass amerikai cég terméke, alapsávú Ethernet típusú helyi hálózat. Újabb változata a Net One Personal Connection, mellyel IBM PC-k is rendszerbe kapcsolhatók. A hálózat egy vagy több szerverállomást (lehet IBM PC XT) és munkaállomásokat tartalmaz, hálózati csatlakoztatását a PC-be dugaszolható hálózati csatolóegység valósítja meg.

A Net One hálózatban élő szerverfunkciók: az osztott fájl- és nyomtatókezelés, valamint egy másik helyi hálózathoz való csatlakoztatást biztosító gateway (kapu) funkció. Nem szükséges, hogy a szerverek „dedikáltak" legyenek; dedikált szerver ugyanis nem tölthet be munkaállomásfunkciókat is a fájlkezelés mellett.

Az alapsávú hálózat koaxiális vagy optikai kábelen 10 Mbps sebességgel működik, a hálózatba kapcsolható állomások száma 64, az áthidalható távolság max. 2,5 km. Létezik egy széles sávú (frekvenciaáthelyezéses) Net One hálózat is, amely sodrott érpárú kábelen, 5 Mbps sebességgel, CSMA/CD protokollal működik; az áthidalható távolság max. 9 km. Mindkét változat MS—DOS operációs rendszer alatt működik. A széles sávú hálózat csatolóegységén kívül a rendszerhez rádiófekvenciás modem is tartozik.

3 Com Ethernet hálózat

A 3 Com Ethernet típusú helyi hálózatában az adatátviteli közeg szintén koaxiális vagy optikai kábel, az átvitel sebessége 10 Mbps, a a bekapcsolható legnagyobb állomásszám 64, a max. kábelhossz 2,5 km és a működtető operációs rendszer ugyancsak MS—DOS. A rendszer Ether Share, illetve Ether Print programjai biztosítják a hálózat működését, valamint az osztott fájl-, illetve nyomtatáskezelést, soros vagy párhuzamos nyomtatóval. Az Ether Share program a hálózat egyik munkaállomásáról tölthető be rendszerindításkor. A hálózati szerverállomás IBM PC XT vagy AT alkalmazása esetén munkaállomásként is használható (nem dedikált szerver), ami a tapasztalatok szerint nem lassítja számottevően a rendszer működését.

A hálózatba archiváló tárként (backupként) 36—252 Mbájt kapacitású szalagegység is bekapcsolható. A rendszer hozzáférési jogosultságát jelszavakkal védik; a lemezterületek vagy közös vagy kizárólagos hozzáférésűek. Az egyidejű írás ellen rekordzár véd.

A 3 Com Ether Mail programja segítségével a hálózatban elektronikus levelezés is folytatható: a névre szólóan elküldött üzeneteket ki-ki lehívhatja saját elektronikus levélládájából.

Omninet hálózat

A gyártó Corvus cég első „helyi hálózata" a Multiplexer volt, amely azonban még nem igazi osztott rendszer, hiszen a többfelhasználós konkurens lemezkezelést a lemezterület előzetes felosztásával (patricionálással) oldotta meg.

Az Omninet max. 2,5 km hosszúságú sodrott érpáron 1 Mbps sebességgel, CSMA/PA hálózateléréssel működő, 64 munkaállomásos rendszer. (A CSMA/PA hálózatelérési módszer hasonlít az 1986. októberi számban ismertetett CSMA/CD-re, azzal az eltéréssel, hogy az adatcsomagot akkor tekinti sikeresen célba jutottnak, ha az ellenállomásról visszakapott egy pozitív nyugtát. Innen száramzik a PA, azaz Positive Acknowledgement megnevezés is.) Az Omninet hálózathoz IBM PC-k, Apple II és III sorozatú gépek, a Texas Instr. professzionális gépei, valamint a DEC Rainbow gépei tartozhatnak, akár vegyesen is. A hálózat előnye az alacsony ár, továbbá hogy különböző operációs rendszerű (DOS 3.1, CP/M/Apple, CP/M—86, UCSD—P) gépek kapcsolhatók vele rendszerbe.

Az Omninet hálózatvezérlő programja a Constellation, melynek I-es változata csakis Apple gépekhez készült, a II-es változat viszont már az IBM PC-ket is kezeli. Nem azonos típusú gépekhez mindenképpen a Constellation II-re van szükség.

A hálózati lemezkapacitás 5—126 Mbájt között változhat, archiválási célokra a szalagegység (200 Mbájt) áll rendelkezésre. A hálózat megbízhatósága igen jó, amit a nyugtázásos hálózat-hozzáférési rendszerrel valósítottak meg (CSMA/PA).

A hálózati szerverállomás dolga az osztott lemezegység- és nyomtatásvezérlés, egykártyás Z80-as vezérlőegységével. Az Omninethez max. 8 szerver kapcsolható, a rendszer hozzáférési jogosultságát a felhasználói név, illetve a jelszó biztosítja.

A központi számítógéppel való kapcsolatot SNA gateway (például IBM PC AT) teremti meg, így az egyes munkaállomások a rajtuk futó IBM terminálemulációs programok — például IBM 3270 — segítségével virtuális terminálkapcsolatot létesíthetnek a központi géppel, és annak termináljaként adatokat vihetnek át, hívhatnak le, illetve programokat futtathatnak rajta. A gateway biztosítja, hogy a fenti szolgáltatást egyidőben több felhasználó is igénybe veheti.

Point 32

A hálózatot három változatban az angol Apricot készíti. A hálózat az Omninet rendszert követi, sintopológiájú, az átviteli sebesség 1 Mbps, a kábel sodrott érpárú, a hálózatelérés CSMA/CD rendszerű, a telepítési távolság 600 m, illetve vonali erősítőegység beiktatásával 1200 m lehet. A rendszerbe kapcsolható munkaállomások száma 32/10 hálózat esetén max. 16 (optimálisan 4-6), a 32/20, illetve 32/80 hálózatban 32 (optimálisan 8-12). A hálózatba max. 10 szerverállomás kapcsolható be. A hálózati szerverek dedikáltak, azaz csak osztott lemez- és nyomtatókezelést végeznek, munkaállomásként nem használhatók (nincs se képernyőjük, se billentyűzetük). A szerver memóriája 512 kbájt, a rendszerhez 10, 20, illetve 80 Mbájtos lemezegység, valamint 3,5"-os floppy tartozik. A kisebb teljesítményű hálózatok háttérkapacitása utólag is bővíthető, 80 Mbájtra.

A hálózatba osztott hozzáféréssel V24, illetve Centronics interfészű nyomtatók (például Epson) kapcsolhatók. A rendszerben tárolt fájlokhoz a felhasználók olvasáskor konkurens módon férhetnek hozzá, az egyidejű írás ellen pedig rekord szintű a védettség. A rendszer egyszerűen bővíthető: új hardver csatlakoztatása esetén új rendszert kell generálni. A hálózatba az Apricot F1, F2, F10, Fp, X_i10, X_i20 gépek kapcsolhatók be.

A hálózatot az MS—NET hálózatvezérlő szoftver — a PC-be dugaszolt csatolókártyán beépítve —, illetve az MS—DOS 3.1 operációs rendszer működteti. A hálózathoz archiválási céllal szalagegység is csatlakoztatható.

PC-net hálózat

Az Orchid Technologies személyi számítógépes helyi hálózata max. 2,5 km hosszúságú, 75 ohmos koaxiális kábelen 880 kbps átviteli sebességgel, CSMA/CD protokollal működik, és max. 64 munkaállomást kapcsolhat rendszerbe. Kisebb eltérésekkel több néven is forgalmazzák: AST's PC-net, Santa Clara System's PC-net, IDE Associates IDEA-net. A hálózati csatoló viszonylag egyszerű felépítésű, kevés hálózati funkciót lát el, ezért a hálózatvezérlés jelentős feldolgozó teljesítményt köt le a PC-ben. Számos opcionális egységet ajánlanak a vevőknek: többfunkciós memóriabővítés, különböző portok, hálózati programbetöltő

kártya stb., melyekkel a felhasználó munkáját könnyítik.

A hálózati szerver egyúttal munkaállomás is, más is. Munkaállomás minden olyan felhasználói PC, melynek perifériái (például a lemezegysége) elérhetetlenek a hálózat többi állomása számára, de lehet osztott hozzáférést biztosító PC is, melynek perifériái „közösíthetők".

A rendszer segítségével megvalósítható a funkciók távoli végrehajtása, például egy programfordítás helyi kezdeményezéssel, de a hálózat egy másik gépén.

A Plan 4000 rendszer

Az amerikai Nestar cég első helyi hálózata a Cluster One volt, amely Apple gépeket szervezett hálózattá. A Plan 4000 koaxiális kábelen működik, 500 m hosszon 254 állomást kapcsolhat rendszerbe, az átviteli sebesség 2,5 Mbps. A hálózathoz fájlszerver, IBM 3270 és 3278 típusú terminálemulációs program, telexkezelő, valamint két Plan hálózatot összekapcsoló — gateway funkciókat ellátó — szerver is kapható.

A fájlszerver Motorola 68000 mikroprocesszor alapú, lemezkapacitása 60 Mbájt. A rendszerhez 500 Mbájtos szalagegység (archiváló tár) is kapcsolható. A tárolt fájlokat jelszó védi. A hálózat munkaállomásai és az őket működtető operációs rendszerek az alábbiak lehetnek: Apple II és III gépek, operációs rendszerük DOS 3.1 vagy CP/M, valamint IBM PC-k, operációs rendszerük MS—DOS.

A Plan 4000 hálózat meglehetősen drága, de van egy olcsóbb változat is, a Plan 3000, amely csak egyetlen — 60 Mbájtos — lemezegységet tud kezelni. A Plan 2000 változat csatolóegysége hordozza az IBM PC XT alapú szervert működtető szoftvert. Mindhárom hálózat összekapcsolható egymással is a gateway szerver segítségével. Két összekapcsolt Plan 4000 hálózat között megvalósítható a virtuális lemezkezelés, azaz az egyik hálózat munkaállomása magához kapcsolhatja a másik hálózat lemezegységét, mintha az fizikailag is az ő gépéhez tartozna.

Net Ware

Az amerikai Novell cég két helyi hálózatot forgalmaz: Net Ware/X-et és a Net Ware/S-et.

A Net Ware/X hálózat 1,43 Mbps sebességű; max. 2,5 km hosszúságú koaxiális kábelt használ, a hálózatelérési protokoll CSMA/CD rendszerű, a hálózatba kapcsolható max. állomásszám 256.

A Net Ware/S 500 kbps sebességgel, 500 m hosszúságú sodrott érpárral dolgozik, a hálózatelérési protokoll egyedi megoldású, a hálózatba kapcsolható max. állomásszám 24.

A Novell hálózatvezérlő szoftvert is szállít különböző gyártók rendszereihez, többek között a Quadnet VI-hoz, a proNET-hez, a Personal Mini-hez, az Omninet-hez, az Ethernet-hez, a PC-net-hez, a Multilink-hez, a Plan 3000-hez, az SMC-net-hez, valamint saját rendszereihez. Az egyes rendszerek közötti egyedüli különbséget a csatolóegység fizikai szintű vezérlése jelenti. A fenti rendszerek majdnem mindegyikében IBM PC XT a hálózati szerverállomás, kivételt képez a saját rendszer, a Net Ware/S, amely Motorola 68000 alapú fájlszervert alkalmaz.

A Net Ware egyedi megoldású rendszer, amelyben különböző trükkökkel sikerült megnövelni a hálózat teljesítményét; például műveletgyorsító előtár (cache tár) beépítésével, illetve a lemezen való ún. elevátoros keresési módszer alkalmazásával, amikor az egyszerre jelentkező fájlhoz fordulásokat úgy sorolják, hogy azokat minimális fejmozgatással lehessen elvégezni.

A Net Ware rendszerhez 15 szerverállomás kapcsolható, amely lehet akár dedikált, akár nem dedikált. A Net Ware operációs rendszere valódi fájlszerver funkciót valósít meg: a fájlok megnyitásakor automatikusan kizárja más állomásoknak ugyanezen fájlhoz fordulását (tehát egyszerre két helyről nem nyitható meg ugyanaz a fájl), továbbá lehetőség van a munkaállomások közötti üzenettovábbításra, valamint az osztott nyomtatásvezérlésre is.

A Net Ware előnyét éppen az adja, hogy a szabványos, és más IBM alapú rendszerekben alkalmazott PC-DOS fájlstruktúrától kismértékben eltérő fájlstruktúrával rendelkezik. Az adatok sértetlenségét és titkosságát igen fejlett adatvédelmi rendszer garantálja: egyrészt a DOS fájlstruktúrájáénál több „zár" iktatható be, másrészt a hozzáférési jogosultságot négy szinten ellenőrzi. Az első szint a logon/password (jelszó), amely a rendszerbe való belépést engedélyezi; a második a trustee (meghatalmazási) szint, amely közli, hogy a felhasználó milyen művelettel fordulhat valamely fájlhoz (írás, olvasás, fájllétesítés, törlés stb.); a harmadik szint a directory right (könyvtári betekintési jog), amely vigyázza, hogy a felhasználó mely könyvtárak fájljait hívhatja le; végül a negyedik szint a file right (fájlműveleti jog), amely megadja, hogy a felhasználó mely fájlokat módosíthatja az adott könyvtárban. Ezeket a jogokat a hálózatfelügyelő rendeli hozzá a fájlokhoz, illetve az egyes felhasználókhoz.

A Net Ware támogatja a dinamikus lemezterület-hozzárendelést, vagyis lemezterület(ek) felhasználó(k)hoz rendelését oly módon, hogy ezek a területek a fájlműveletek (például törlés) függvényében módosulnak.

A szerverhez max. három nyomtató — két soros és egy párhuzamos — kapcsolható, a nyomtatás a fájlkezelés mellett háttérműveletként hajtódik végre. A Net Ware nyomtatási rendszere az egyik legjobb; olyan opciói vannak, mint a többpéldányos nyomtatás, nyomtatás közbeni papírváltás, nyomon követhetők a függőben lévő nyomtatások, átrendezhető a nyomtatás várakozási sora, a várakozó nyomtatások egyik nyomtatóról a másikra átirányíthatók. Mindezek a speciális nyomtatásvezérlő utasítások segítségével hajthatók végre. Meg kell említeni azonban, hogy a Net Ware — számos előnye mellett — nem képes néhány DOS parancs végrehajtására. Ilyenek például: PATH, CADIR, BACKUP.

Nem „igazi" helyi hálózatok

A tapasztalat szerint meglehetősen gyakoriak a szerényebb felhasználói igények, különösen kisebb intézmények részéről. Az olcsóbb hálózatok rendszerint csillag topológiájúak, és egy központi gép köré települnek, amely időosztásos rendszerben működik, és szabványos, például V24 interfészen keresztül tart kapcsolatot 4-8 munkaállomással. Munkaállomásként egyszerű terminál is megfelel, ami jelentős árcsökkentést tesz lehetővé.

Az ilyen típusú rendszerek központi gépe akár egy IBM PC XT vagy AT is lehet, és például az Anex cég által készített hálózati interfészkártyával a rendszer viszonylag olcsón négyfelhasználóssá tehető. Ezt a rendszert az Anex Multi DOS operációs rendszere működteti oly módon, hogy a műveletvégző teljesítményt megosztja a központi gép és a munkaállomások között.

Ezek a rendszerek alkalmasak olyan közismert szoftvertermékek futtatására is, mint a Multimate vagy a Lotus 1—2—3, és képesek lebonyolítani felhasználóik elektronikus levelezését. Tapasztalatok szerint a 3-4 felhasználóval működő hálózat gépi válaszidői alig hosszabbak a hagyományos helyi hálózatok válaszidőinél. Nagyobb távolságok áthidalásához természetesen modemre és adatátvitel-vezérlő szoftverre is szükség van.

Ezeket a rendszereket elsősorban olyan alkalmazásokhoz ajánlják, ahol az adatállományt központilag tárolják, de több helyről, nem feltétlenül egyidejűleg kérdezik le, illetve aktualizálják. Az ilyen rendszerekben tehát nem alapkövetelmény, hogy minden állomás egyszerre férhessen hozzá a teljes adatállományhoz. Két ilyen megoldást mutatunk be.

a) Multilink Advanced

A Davong cég által fejlesztett rendszer a Multilink Advanced hálózat, amely kilenc állomás számára biztosít többmunkahelyes (multi user) és többfelhasználós (multi tasking) környezetet. A hálózat gépeit MS—DOS operációs rendszer működteti; a rendszer nagy előnye, hogy működéséhez nincs szükség drága hálózati csatolóegységre. A Multilink rendszer központi gépe IBM PC AT. Ennek feldolgozó teljesítményén osztoznak a munkaállomások.

Megfelelő memóriabővítéssel nagyobb tárigényű programok futtatása is megoldható. Ez esetben egy-egy felhasználóra 448 kbájt központi tárkapacitás jut. A rendszerhez mind hardcopy nyomtató, mind osztott hozzáférésű „központi" nyomtató kapcsolható (soros vagy párhuzamos interfészen keresztül), melyet a hálózat egyes állomásai bejelentkezésük sorrendjében használhatnak. A fájlkezelési rendszer osztott hozzáférést biztosít a központilag tárolt programokhoz, illetve adatállományokhoz. A fájlvédelem jelszavas.

A Multilink hálózaton olyan közismert programtermékek is futtathatók, mint a Word Star, a dBase III, a Multimate és a Lotus 1—2—3. A munkahelyi interfész V24, és olcsó terminálokkal a hálózat teljes ára nem éri el egyetlen AT gép árát. Az alacsony ár a rendszer egyik legnagyobb előnye.

A hálózat munkaállomásai többek között DEC VT 52, VT 100, illetve IBM 3101 gépek lehetnek. A rendszerhez több cég szállít interfészegységet, például az IBM (Communication Adapter), a Software Link (Multiuser Communication Port Board) vagy a Quadron (Dual Assynchron Card). A terminálhálózat prioritásos rendszerűre is kiépíthető.

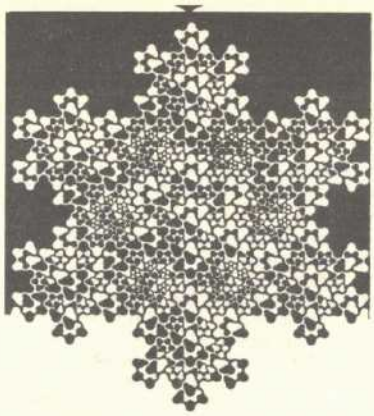
b) A LAN Link hálózat

A Multilinkhez hasonló egyszerű hálózat a LAN Link, amely egy központi telepítésű IBM PC-ből (célszerűen AT) és a hozzá csillag elrendezésben, V24 interfészen keresztül kapcsolódó négy munkaállomásból áll. Az olcsón megvalósítható hálózat központi számítógépét, a szervert, valamint munkaállomásait PC—DOS 2.0, MS—DOS 2.0 verziószámú vagy valamely fejlettebb operációs rendszer működteti. A hálózat átviteli sebessége 112 kbps. A hálózat munkaállomásai adott esetben közönséges képernyős terminálok is lehetnek, de természetesen IBM PC-k is.

A hálózatba soros, illetve párhuzamos interfészű nyomtató kapcsolható, amely a hálózati munkaállomások számára biztosítja az osztott nyomtatást. A munkaállomások közvetlenül vagy modemen keresztül csatlakoztathatók a szerverre. A fájlkezelő rendszer jelszóval védett, és megengedi a fájlokhoz való egyidejű hozzáférést olvasásra, de az egyidejű írás ellen rekord szintű védelmet ad.

CSEH KÁLMÁN

FANTÁ
A Mandelbro

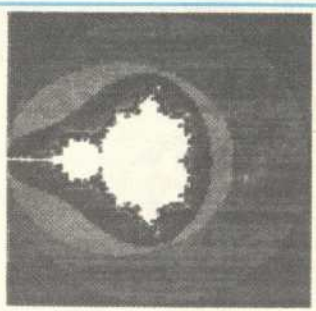


A Mandelbrot halmaz egy rendkívül bonyolult alakzat, melyet a $Z_0=0$, $Z_{i+1}=Z_i^2+C$ rekurzió definiál. A Mandelbrot halmaz elemei azok a C komplex számok, melyekhez tartozó $|Z_n|$ sorozatnak van véges felső korlátja. Bizonyítható, hogy ha egy k-adik iterációs lépésnél $|Z_k|\geq 2$ értéket kapunk, ennek a $|Z_n|$ sorozatnak nem lesz véges felső korlátja, vagyis az ehhez tartozó C komplex szám nem eleme a Mandelbrot halmaznak.

A PTA—4000 típusú gépre készített *1. programmal* a komplex számsík egy téglalap alakú részét vizsgáljuk, hogy ennek pontjai elemei-e a Mandelbrot halmaznak, vagyis C értékei e tartomány pontjai lesznek. A program egy C számot akkor tekint a (közelítő!) Mandelbrot halmaz elemének, ha az általunk megadott maximális iterációszámig a rekurzív sorozatban egyetlen $|Z_i|\geq 2$ értéket sem kap (az ábrázolásnál ezekhez a pontokhoz nem rendel színt). A tartomány más pontjaihoz aszerint a K

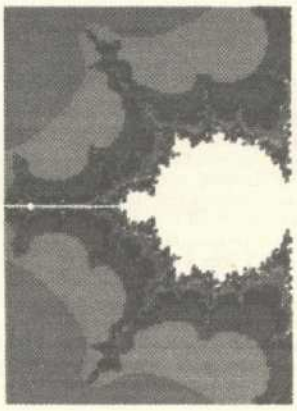
1. program

```
MANDELBROT 1

10:INPUT "X0=";X, "X1=";Q, "Y0=";Y, "Y
   1=";U
20:INPUT "ITERACIOK (>15)  =";N
30:IF N<15GOTO 20
40:INPUT "COLOR MOD (0-2) =";M:POKE
   12418,M:POKE 12478,56+157*(M)0),
   56+157*(M)1)
50:F=(Q-X)/100:G=Y-F:H=4:J=(U-Y)/P
60:FOR I=0TO J:CALL 12318:CALL 1248
   7:NEXT I
80:TEXT :CSIZE 1:COLOR 8:LF 3:
   LPRINT "REAL PART :";X;:LCURSOR
   17:LPRINT ",";Q
90:LPRINT "IMAG.PART :";Y;:LCURSOR
   17:LPRINT ",";U
100:LPRINT "COLOR MOD :";M
110:LPRINT "ITERACIOK :";N
120:LF 4:END
```



```
REAL PART :-2     , 2
IMAG.PART :-2     , 2
COLOR MOD : 0
ITERACIOK : 30
```

2. program



```
REAL PART :-1.5   ,-1.25
IMAG.PART :-0.18  , 0.18
COLOR MOD : 2
ITERACIOK : 80
```

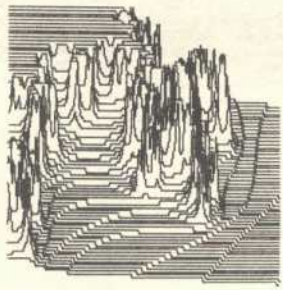
```
REAL PART :-0.864,  0.864
IMAG.PART :-0.35 ,-0.76
COLOR MOD : 1
ITERACIOK : 60
```

ÉS TUDOMÁNY?

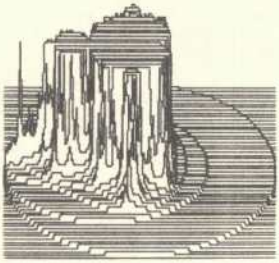
maz előállításakor keletkező
színes" ábrák

(Hátsó borítónkon)

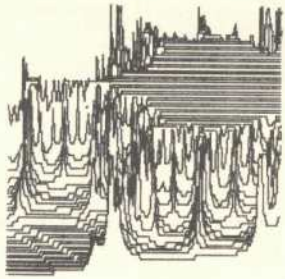
```
MANDELBROT 2

10:INPUT "XB=";X,"XJ=";J,"YB=";Y
15:INPUT "N=";E:POKE 17418,E
20:GRAPH :F=(Q-X)/216:H=4:W=J*256+
   254:U=216*F+Y
25:Q=(X+Q)/2:U=(Y+U)/2:POKE 17476,1
   54
30:INPUT "PHI=";O,"THETA=";B:M=SIN
   O:K=SIN B:D=COS B:N=COS O
40:GLCURSOR (188,-255):SORGN
50:A=E*D+188*(K*N+M*K):S=-188*(K*N+
   M*K):U=255/(A-S)
60:FOR I=-188TO 188STEP 4:G=U+1*F
80:FOR J=-188TO 188:T=(J+1*M)/N:Z=Q
   +T*F
90:CALL 17348:Z=INT ((D*PEEK W-J*M*
   K+1*N*K-S)*U+.5):R=PEEK (W-183+J
   )
100:IF J=-188GLCURSOR (J,Z):E=(Z/R)
105:IF I=-188GOTO 150
110:IF R)=ZAND E=0GOTO 160
120:IF R)=ZLET E=0:Z=R:ON (X<>R)+1
   GOTO 150,160
130:IF E=1GOTO 150
140:E=1:GLCURSOR (J,R-(R-Z)*(X>=Z))
150:LINE -(J,Z):POKE W-183+J,Z:R=Z
160:X=R:NEXT J:NEXT I:POKE 17476,165
   :GLCURSOR (-188,-18):TEXT
```



XB, XI=-0.8648, 0.8648
YB=-0.9188
N= 38



XB, XI=-2.8888, 2.8888
YB=-2.8892
N= 38



XB, XI=-1.5888, -1.2588
YB=-0.1255
N= 38

index (ezt „kiesési index"-nek nevezzük) szerint rendel egy-egy színt, amelyiknél először kap $|Z_k| \geq 2$ értéket. Természetesen minél nagyobb maximális iterációszámot adunk meg, annál pontosabb képet kapunk a Mandelbrot halmazról.

A bemenő adatok (baloldalt a változó neve, jobboldalt az értelmezése):

X_0, X_1	A vizsgált tartomány valós részének kezdő- és végpontja;
Y_0, Y_1	A vizsgált tartomány imaginárius részének kezdő- és végpontja;
ITERÁCIÓK	Az általunk megadott maximális iterációszám (maximális értéke 255 lehet), ameddig a vizsgálat folyik;
COLOR MOD	Értéke 0, 1, 2 lehet. A Mandelbrot halmazhoz nem tartozó pontokhoz rendelt szín a következőképpen lesz meghatározva:

$$\text{INT}\left(\frac{\text{K kiesési index}}{2^{\text{color mod}}}\right) \pmod 4$$

ahol INT jelöli az egészrész függvényt.

0 választásnál más-más kiesési indexhez más-más szín jelenik meg, 1 választásnál minden második, 2 választásnál minden negyedik kiesési indexnél jelenik meg más szín, így elkerülhetjük az ábrák esetleges túlzott tarkaságát.

A program az ábra elkészítése után kinyomtatja az általunk megadott adatokat is. Az ábra a legfinomabb felbontás négyszeresével készül (X és Y irányban is kétszeres felbontás), a sebesség növelése érdekében.

A *2. program* a komplex értelmezési tartományt az XY sík részeként, a kiesési indexet pedig ezekre merőleges térbeli Z — nem komplex! — tengelyen ábrázolja. (A

```
GÉPI KÓDÚ RÉSZ : 4310 h-4498 h

4310 >   BE 43 50 BE EF 8A 8A 43
4318 >   7E BE 43 50 BE EF 86 8A
4320 >   43 7E BE 43 50 BE F8 1A
4328 >   8A 43 7E 50 79 48 79 D9
4330 >   D9 D9 1A 84 D9 D9 D9 8A
4338 >   6A 87 E5 88 83 8A 5A 88
4340 >   8E 82 5A 18 58 7A 48 79
4348 >   D9 D9 D9 8A 6A 87 F5 88
4350 >   83 9A AE 47 FF 14 D9 D8
4358 >   D9 1A 58 78 84 D9 D9 D8
4360 >   8A 48 78 F0 98 58 7A 5A
4368 >   18 6A 87 F5 88 83 FD 8A
4370 >   5A 88 6A 87 F5 88 83 9A
4378 >   A5 47 FF 58 79 D9 D9 D8
4380 >   1A 48 7A 4A 88 6A 87 F5
4388 >   88 83 9A B5 81 FD C8 8E
4390 >   43 88 FD 8A 48 47 8A A5
4398 >   47 FE 41 84 B7 65 33 13
43A0 >   9A B5 86 5A 86 4A B5 BE
43A8 >   43 16 B5 18 4A 17 BE 43
43B0 >   31 BE 43 31 B6 B5 16 5A
43B8 >   18 4A B5 BE 43 16 B5 81
43C0 >   4A 86 BE 43 31 B5 88 4A
43C8 >   18 BE 43 31 E8 47 FE 88
43D0 >   B5 8E 4A 88 5A 88 BE 43
43D8 >   28 B5 8F 5A 81 4A B1 BE
43E0 >   43 28 B5 88 5A 8E 4A 8F
43E8 >   BE 43 16 B5 88 BE 43 44
43F0 >   B5 87 BE 43 48 B5 83 BE
43F8 >   D8 D2 88 42 EF 47 FE 81
4400 >   A5 47 FE 87 1E 88 81 9A
4408 >   B5 82 5A 8E 4A 8F BE 43
4410 >   1F B5 82 5A 82 4A 18 BE
4418 >   43 16 B5 88 5A 88 4A 88
4420 >   BE 43 16 B5 81 5A 88 4A
4428 >   81 BE 43 28 B5 81 5A 81
4430 >   4A 86 BE 43 16 B5 88 4A
4438 >   82 BE 43 31 8E 6F A5 47
4440 >   FE 38 38 B8 83 AE 47 FE
4448 >   9A 4A 82 BE A2 C4 6A 88
4450 >   FD A8 BE A8 E4 BE A5 18
4458 >   BE A7 60 E8 47 FC 88 48
4460 >   47 4A 81 FD 2A 24 87 88
4468 >   18 FD 88 FD A8 A5 47 FC
4470 >   8A BE A2 8E B5 82 BE AC
4478 >   2F BE A7 60 E8 47 FC 88
4480 >   FD 2A FD 8A EF 47 FC 82
4488 >   48 4E 65 98 28 68 6E 84
4490 >   99 42 9A

Beírás : POKE-vel

KAZETTÁRA RÖGZÍTÉS :
CSAVEM "név",17174,17568

BETÖLTÉS KAZETTÁRÓL :
CLOADM
```

— közelítő — Mandelbrot halmaz pontjaihoz az általunk megadott maximális iterációszámot rendeljük.) A szemléletesebb és esztétikusabb ábrázolás kedvéért lehetőség van a vizsgált tartomány középpontja körül phi szöggel az YX síkot, theta szöggel az XZ síkot elforgatni.

A bemenő adatok:

X_0, X_1	Az ábrázolandó tartomány valós részének kezdő- és végpontja;
Y_0	Az imaginárius rész kezdőpontja (a végpont mindig $Y_1 = Y_0 + X_1 - X_0$ lesz, vagyis mindig egy négyzet alakú tartománnyal dolgozunk);
N	Az általunk megadott maximális iterációszám;
phi	XY sík forgatásának szöge (a példákban 0);
theta	XZ sík forgatásának szöge (a példákban 0,18-0,22 radián).

A gépi kódú részt mindkét program használja. Bővítő nélküli gépeknél vigyázzunk, nehogy javítás, beszúrás, ill. más ok miatt a BASIC program a gépi kódú program által lefoglalt memóriaterületet megváltoztassa. Ajánlatos ezért a gépi kódú rész néhány első bájtját ellenőrizni és esetleg javítani.

MÉSZÁROS CSABA

Olvasóink figyelmébe ajánljuk Peitgen H.O.— Richter P.H The Beauty of Fractals Images of Complex Dynamical Systems (Springer 1986) c. könyvét, amely ebben a témában elméleti és rendkívül gazdag képanyagával kiemelkedő összefoglaló munkának számít.

Mottó: Az igazság mindig egyszerű.
(Egy középiskolai tanártól)

Barátunk, a számítógép

Együtt dolgozunk

Alig két éve majdnem írtam egy cikket *Mikrogép és aerobic* címmel. Akkor azt hittem, hogy azonos szintű, múló divatokról van szó. Tévedtem. Az aerobic ment, a mikró maradt. Ne feledjük azonban az aerobic szorgalmazójának, Jane Fondának az infarktusát! Példája mutatja, hogy egészséges dolgokkal is lehet visszaélni, mint például a mikrókkal! Azt is tartsuk vigyázó szemeink előtt, hogy a fejlett számítástechnikájú országokban a mikró nem kíván az lenni, amilyen szerepet betölt nálunk a számítástechnika-éhség miatt. Ott a mikró nem pótolja, hanem kiegészíti a nagygépes lehetőségeket. Már számtalanszor előfordult velünk, magyarokkal, hogy a más által kitalált lehetőséggel sokáig nem éltünk. Majd ezután mi megmutattuk, túlhajtva az eredeti célon. Ettől talán óvakodnunk kellene!

A mikró fogalma rendkívül eltérő kapacitású, teljesítményű és általános lehetőségű gépek széles skáláját takarja. Jól hasonlítható egymással egy régebbi típusú IBM nagygép és egy újabb. Ugyanakkor egy „zseb"-Sinclair képességeit aligha lehet összevetni egy IBM AT lehetőségeivel, nem beszélve a 32 bites, szekrénnyi IBM RT-ről. Ezért a további mondanivalót kényszerűen a közepes kategóriájú (IBM PC XT, AT és hasonlók) mikroszámítógépekre kell korlátoznom. Megjegyzendő, hogy e kategória alatt (például Commodore 64) a lehetőségek drasztikusan csökkennek, míg a magasabb kategóriájú gépek hazai elterjedésére még várnunk kell.

Ha a barátunk használatával kapcsolatos tapasztalatainkat és érzéseimet összegeznem kellene, akkor mondanivalómat három pontba illene összefoglalnom. A korábbi számítógépes lehetőségekkel és azok hiányával szemben a mikrók világa mást jelent a végső felhasználónak és mást a fejlesztőnek, a számítástechnikai szakembernek. Ugyanakkor a mikró — mikró. Olyan általános jellemzőkkel is leírható — a nagygépekkel szemben —, amelyek mind a felhasználó, mind a fejlesztő lehetőségeit meghatározzák. Logikusnak tűnik, hogy mondanivalómat ezen általános jellemzőkkel — korlátokra és lehetőségekre bontva — kezdjem. Ezt a rövid összefoglalót a végső felhasználók figyelmébe ajánlom. Annak ellenére, hogy itt bizony néhány technikai részkérdésre is ki kell térnem. Kezdjük tehát az általános lényeggel!

Gyermekbetegségek

Előttem egy IBM PC XT, 640 kbájt központi egységgel és 20 Mbájt háttérkapacitással. Jobbra egy IBM PC AT 3 Mbájt központi egységgel és 80 Mbájt háttérkapacitással. Mennyi ez?

Nem kevés! Korábban a jó öreg IBM 370-en dolgoztam, amely 750 kbájtos központi egysége mellé hiába rendelkezett igen bő háttérkapacitással, abból én csak 52-100 Mbájtnyit használhattam ténylegesen. (Csak a kevésbé tájékozottak kedvéért jegyzem meg, hogy egy szabványos gépelt oldal (A4) 1800 karaktert (bájtot) jelent. Egy 500 oldalas könyv tehát bőven elfér 1 Mbájt területen.) Közepes végső felhasználóként. Mit jelent a „közepes"? Nos, volt vagy száz programom és mintegy 150 ezer nyilvántartási egységem, rekordom.

Mindez azt jelenti, hogy a mikrók adott kategóriája a hozzám hasonló „közepes" (magyarországi viszonylatban!) felhasználók számára akár meg is felelhet. Azonban azt a 20 Mbájt háttértárat „pillanat alatt" megeszik a programok, és a 80 Mbájtból is csak falatnyi marad, ha elkezdjük az adatok tényleges betöltögetését.

Nem, kis barátunk határozottan nem való komoly, összetett vállalati rendszerek támogatására. Legfeljebb integrálatlan részrendszerek kezelését biztosítja számunkra, ha helyesen választottuk meg a gép egységeit, az ominózus konfigurációt.

Kiváló kollégáim közül némelyek most ráncolhatják homlokukat: ők találtak megoldást. Nos, a találmány így hangzik: a rendszert részekre kell szabdalni. Ha az egyik ikszedével végeztünk, akkor kimentünk, betöltjük a második ikszedet és minden megy a maga sorjában. A szegény ember főztje ... A 60-as években dolgoztunk így, lekapkodva-felrakosgatva a mágnesszalagokat. A helyzet vészesen hasonló. Akkor ezt számunkra az operátor végezte, tiszteletreméltó számú kilométereket téve meg a konzol és a szalagegységek között.

A 80 Mbájtból is csak „falatnyi" marad

Ritkán, de zavarba jött; mit is kellene most fel és mit le. Most ezt magunk élvezhetjük. A kimentés és visszatöltés pedig nem egy pillanat. Ne adj Isten, hogy közben átszervezésre is igény legyen!

Nyilvánvaló, hogy ez a módszer adott esetekben — összefüggő állományok és programok esetében — nem is használható. Az esetek másik részében jön a sakkozás: mit, mikor és hová. Mindezzel pedig nemcsak a kimentés-betöltés idejét kell eltűrnünk, hanem egyéb teljesítménycsökkenésekkel is kell számolnunk.

Oda se neki! Magyarországon még egyetlen vállalat sem ment tönkre abban — másutt már százával —, hogy egy információ percekét késett. Valóban: nekünk rendkívüli élmény a feldolgozási lánc lerövidülése, az említett gyalog módszer kellemetlenségei dacára is online elérhető információ. Nem kell formanyomtatványra írnom az adatot. Talán még kódoltatnom sem kell. Nem lyukasztja félre információim x százalékát az adatrögzítő. A hibát azonnal látom és azonnal javítom. Nem kell gépidőért könyörögnöm. Nem kell futás után a „kimenőben" keresgélnem az anyagot. Mindez remek!

Csak néhányan már láttak nagygépes online rendszert is! Nem érzékeltük, hogy adatunk mentődik-betöltődik-e vagy sem. Nem kapkodtunk diszkettcserék után. Szóval a mikrogépes rózsa nem csak illatozik, hanem nemesítettekhez képest a kelleténél kicsivel jobban tövises. Sokat lendítene a helyzeten a cserélhető rögzített lemez. Még nincs.

Tudom, hogy ej, ráérünk. Ezért csak közbevetőleg jegyzem meg, hogy hiába van nekem szép központi táram. Egy komolyabb nagygépes lemezegység gepárdjához képest az én kis rögzített lemezem egy csiga. Húsz adatnál, ötvennél, száznál? Kit érdekel mindez? Egy több ezres vagy több tízezres állományból való szelektív visszakeresésnél már kaparom a kezelőpultot: no, gyere már!

Mert itt van, ugyebár, az adatkezelés nem elhanyagolható kérdése. A háttértárak technikai adottságain kívül a sebességet nagyjából két dolog határozza meg: a tőlem függő adattömeg és az adatkezelő. No, itt mutatkoznak a gyermekbetegségek!

Az adott kategóriába tartozó gépeken rendelkezésünkre állnak a — nagyon régen — magas szintűnek nevezett nyelvek (COBOL, FORTRAN, PL/1 stb.). Ezek azonban csak igen korlátozott állománykezelési képességekkel bírnak. Ha ravasz megoldásokkal magunk láncolunk, indexelünk, mutatunk, akkor elérjük a kívánt gyorsaságot. Az első változásnál beleőrülünk a módosításba. Ha pedig valóban magas szintű adatkezelőt alkalmazunk (dBase III, dACCESS, rBase stb.), akkor el kell tűrnünk minden általánosított rendszer hátrányát: a viszonylagos lassúságot. Mi tagadás, egyelőre még a mikrók nem olyan jól ellátottak hatékony adatkezelési lehetőségekkel, mint nagyobb rokonaik.

A relatív lassúsággal meg lehet alkudni, és ezt könnyen megteszi a magyar felhasználó. A végső felhasználó, aki nem igényel valóban percre pontos információt vagy — eddigi tapasztalat hiányában — nem tudja, hogy ilyet igényelhetne. A fejlesztő, aki eddig a lyukasztóban üldögélt órákat és most percek alatt ütheti be a programját. Melyikük fog a gyorsaságra panaszkodni?

A gyorsaság csak az egyik fontos kritérium a számítógép alkalmazásánál, talán nem is a legkiemelkedőbb. Voltaképpen az időtől függ össze a biztonság kérdése is. Mégpedig húsbavágóan komolyan. Éppen csak a minap „állt fejre" egy nem általunk üzemeltetett rendszer, amelynek pontos időben igen fontos pénzügyi információkat kellett volna biztosítania. Nem volt „back-up", azaz háttér; azért az adatok tíz százaléka megmaradt.

Mi szükséges a valódi biztonsághoz? Két dolog. Egyrészt a háttér. Programok esetében ez nem ügy. Készítünk róluk másolatot és szükség esetén visszatöltjük azokat. A program nem változik, ha nem szándékosan módosítjuk. Más az adat, amelynek tartalma naponta módosulhat, hiszen éppen ez az online rendszerek értelme. Mit naponta? Óránként, sőt percenként! Ha már itt van egy közvetlenül elérhető számítógép, akkor feltétlenül cél, hogy mielőbb bevigyük azt az ismeretet, amit megszereztünk.

Adott mértékű változások, adatbevitelek, törlések után — hogy hol a mérték, azt a rendszer jellege mutatja a tervezőnek — másolatokat készítünk állományainkról. Ez idő, és egyéb feldolgozásoktól von el teljesítményt. A mikrón érzékelhető módon. Nagygépen mindezt nem is észleljük. Ha nem feledkezünk el erről a követelményről, akkor félig rendben is vagyunk.

Félig. Mert pillanatonként, minden feladatunk után nem készíthetünk háttérmásolatot. Végeztünk, mondjuk harminc módosítást — ötvennél akartunk másolatot készíteni — és puff, elszállt a rendszer. Mi a teendő? Akár végső felhasználó, akár fejlesztő vagyok, be fogom tölteni a korábbi, nem sérült változatot. Majd eszméletlen nyomozásba fogok, hogy a korábbi változat óta miket is csináltam?

A mikrogépes „rózsa"

Most mindazt újra elvégezhetem, ha még emlékszem a dolgokra. Vezessek talán róluk kézi naplót?

Nos, tegyük fel, hogy úgy döntök: gépi naplót vezetek. Rögzítse csak a gép, hogy mikor és mit végeztem. Majd egyszerre csak azt észlelem, hogy így az átlagos választidő megkétszereződött. Nem írhatom kis barátom rovására ezt: kétszer jegyezte fel mondanivalómat. És mindettől nem leszek boldogabb, mert elő kell keresnem, amit addig végeztem, és újra végre kell hajtanom. Egy nagyszámítógépes rendszer esetén ilyen bajaim nincsenek. Csak elmaradt környezetben nincs automatikus, a feldolgozási időt gyakorlatilag nem befolyásoló naplózás. Csak közepesen elmaradott környezetekben nincs automatikus újraindítás és a korábbi tranzakciók automatikus újra végrehajtása. Szóval: feladatomtól függ, hogy szabad-e mikróban gondolkodnom, ha megfelelő biztonsági szintet és reális választidőt együtt kívánok tartani.

És végül itt van egy lényeges momentum. Te ülsz a gép mellé vagy én? Tudomásul illene venni, hogy a személyi számítógépek nem véletlenül viselik ezt a nevet! Jelenleg két alternatíva mutatkozik számunkra, a fenti kérdés — ki ül a gép mellé — megvitatásán kívül.

Az egyik lehetőség: egyikünk se. Végső felhasználók vagyunk, és van, aki elvégzi a technikai feladatot. Kezeli a gépet. Ekkor azonban következik a formanyomtatvány, a kódolás, a hibalista, a forduló, a gépidőbeosztás problémája. Kis gépen imitáljuk a nagy gondjait. Nem egy kiváló megoldás.

A másik lehetőség: használunk valamilyen osztott feldolgozási módszert. Ilyesmi az operációs rendszerek szintjén igen korlátozottan megoldható. Ha azonban egyidejűleg ugyanazokat az adatokat akarjuk kezelni, akkor vennünk kell egy erre a célra alkalmas adatkezelőt. És akkor látjuk, hogy ilyen nincs. Ha, korlátozott felhasználót kiszolgálva, mégis találunk valamiféle szoftverszerkentyűt, akkor egyszerű képletbe botlunk: jé, ketten vagyunk felhasználók; és mindegyikünk feldolgozási ideje a háromszorosára(!) nőtt!

Mi tagadás, kis barátunk még nincs ellátva az állományok osztott használatát biztosító képességekkel. Óva intenék bárkit attól, hogy egy ilyen „többfonalú" rendszert maga fejlesszen ki, hacsak nincsen pár tízmillió dollárja, vagy nem hagyja hidegen az esetleg be nem fejezett és vissza nem fejtett tranzakciók sorsa. Vagyis ez az osztott használat szorosan összefügg a korábban feszegetett idő és biztonság kérdésével. Imitáljunk tehát egy régi vágású magyar nagyszámítógépes környezetet, sorban álló felhasználókkal, egy kis gépecskén?

A gondok nem elhanyagolhatók. És még nem is szóltunk a valóban „távfeldolgozási" feladatokról. Ilyesmik már megvalósíthatók mikrókon is, ha valaki az idő és a biztonság említett korlátait eltűri.

A BMW vállalatnál egy tranzakcióbevitelünk — az adatok megfogalmazásától a végeredmény elkönyveléséig — mintegy 20 másodpercet vett igénybe. Ezalatt a számítógép több mint ezer(!) egyéb feladatot hajtott végre. Miközben teljes biztonságban éreztettük magunkat, és adatainkat mások is kezelték. Mire megyünk egymással, kis barátom? Hiszen teljesen világos, hogy ha még osztott használatú, kényelmesen biztonságos, türelmemet próbára tevő sebességű feladatok elvégzésére is képtelen vagy! Miért és miként lehetsz mégis barát?

Ne haragudj, de most le kell szögeznem, hogy semmi olyasmit nem tudsz, amit egy komoly nagygépes online rendszer ne tudna! Lásd be, hogy nekünk kevesebb az anyagi forrásunk, és ezért melegen bízunk a te barátságodban. Lássuk csak, hogy szerény áron mennyi újjal örvendeztettél meg bennünket.

Most végsőfelhasználók

Tegyük fel, hogy most én vagyok a legvégsőbb felhasználó vállalatunkban! Ide került kis barátunk, a mikrogép, és én éppen lelkesedni fogok érte, vagy pedig legyintek. E két, szélsőséges magatartás között is ingadozhatok. Lássunk példákat!

1. példa

Jártak nálunk — ismered őket —, amolyan szervezők vagy mik. Szóval programozók, tudod. Kéjszítettek egy valamit, amit készletnyilvántartási rendszernek neveznek. Ott van azon a vacakon, azon a dobozon, azaz a számítógépen. A főnök behívott és a kezembe nyomott egy olvashatatlan sokszorosítmányt. Tanulmányozzam, majd mindent megértek. Utána a szakemberek elő is fognak adni nekem arról a rendszerről. És január 1-jétől — ezek a dinnye szervezők miért mindig pont ezt a lehetetlen napot jelölik ki átadásra — ezzel a géppel fogok dolgozni, állítólag.

Hát, tudod, én elolvastam azt a valamit. Kódoljak így és használjak olyan betűket. Nehogy véletlenül az x után y-t csináljak, mert az baj, azt a gép nem tűri. Legjobb, ha előre papíron leírom az eseményeket, a kézikönyv előírásainak megfelelően, majd beviszem. A gépbe. A piros hármas gombot kell lenyomnom, ha a zöldségeket, a sárga ötöst, ha a karalábét akarom megismerni. Bal kisujjammal tartsam lenyomva az Alt és Ctrl — érted Te ezt? — billentyűket egyidejűleg, miközben jobb mutatóujjammal Z-t ütök és orrommal lazán érintem a felső állású F-et. Majd RETURN — tudod, ilyen a balatoni vonatjegy is — nyomandó, ha . . . És PgUp, ha felfelé akarok „péjdzselni". (Te, szerinted az mi?)

Hát tudod mit fogok én nyomni, ha a főnök tényleg bevezeti ezt a rendszert? Az ágyat, sokáig. Remélem, majd csak elviszi innen az ördög ezt a frászkarikát. Addig is ölel szerető barátod. (Ui.: Hallottam, utálod a Kovácsot. Ajánld neki ezt a rendszert!)

2. példa

Cicuskám! Ez itt egészen franko! Az a mókus a SZÁM . . .-ból egy fél délelőtt itt magyarázott. Bekapcsolt egy olyan kis masinát, amit a tévében is mutattak. Pár vacak billentyűről — talán ötről — elmondta, hogy mit csinálnak. Kicsit túlmagyarázta a dolgot. Beteg-nek néz ez bennünket? Ne tudnék megjegyezni francos öt billentyűt?

Tudod, egy készletrendszerről van szó. Ez a kis vacak mindent megcsinál. Nem kell papírra írogatni, csak egyszerűen beütni olyan írógépen. Mellettem van egy csinos doksi — ezt Te, drágám, nem érted, a dokumentáció, amely röviden mondja el nekünk, hogy mit és hogyan csinájunk —, amit bármikor fellapozhatok. Tudom, hogy abba a fényes dobozba csak A-t, B-t és C-t ütheték — írja a doksi. Ha elnézem, a gép szeliden és okosan figyelmeztet. Egy pillantás a doksiba, egy a képernyőre, és minden megy.

Kár, hogy kicsit szűkszavú. Azt mondja, hogy rossz adat. De melyik és miért? No, látod, Parányom, ezért van a doksi. Fél perc alatt megtudhatom, hogy mit toltam el.

Szóval így, drágám. Persze azért Rád is gondolok. Ha visszajössz, Neked is megmutatom, olyan izgi. A hétvégét, ugye együtt töltjük? Addig is csókollak.

3. példa

Kedves Bátyám! Ismered a Szabót, aki nálunk számítógépekkel foglalkozik. Képzeld, azt állította, hogy készített egy készletnyilvántartási rendszert. Egészen egyszerű — állította. Így kapcsold be a gépet, és utána csak annyit mondj neki, hogy KÉSZLET. Onnan minden magától megy. Nem sokat beszélt róla, olyan szűkszavú. Lehet használni, ha akarjuk — így ő.

Nem akartam ezt elhinni. Mikor a többiek elmentek, bementem abba a szobába, ahol az a kis gép van. Úgy tettem, ahogy Szabó mondta: bekapcsoltam a gépet. Minden simán ment. Már láttam ilyen gépeket, tudod. Azt is, hogy egy kis angoltól nem esem kétségbe. Itt azonban nem azt mondta nekem a gép, hogy „enter date", hanem azt, hogy jelezzem a dátumot. És elfogadta a magyar dátumot. Majd nem a szokásos — másutt tapasztalt és érthetetlen — speciális jel villant fel, hanem ez a kérdés: Melyik rendszert kívánja használni?

Gondoltam, megpróbálom. Hangsúlyoznom kell, hogy semmit sem tudtam az egészről. Mondd neki: KÉSZLET — így említette Szabó. Hát beírtam: KÉSZLET.

Tömör és udvarias módon kérdezte tőlem a gép: miről kívánok tájékozódni? Akarok-e azonnal programot futtatni? Nem, azt nem, nem is tudom, miről van szó. Akarom-e megismerni a rendszer tartalmi lényegét, vagy inkább technikai használatának módjával kívánok ismerkedni? Nem árt egyik sem, másik sem — gondoltam, és az egyiket választottam. A gép mindent elmagyarázott a rendszerről és megkérdezhettem, hogy mit is ért „átlagos ár"-on. Megmondta. Majd kedvem támadt átváltani a

„. . . No, gyere már!"

technikai kérdésekre. Erre a gép elmagyarázta, hogy így adok be adatot, úgy kapok eredményt, ha hiba van, mit kell csinálnom. Ha valamit nem értettem, mindig lenyomhattam a „H" vagy az „F1" billentyűt — ezt sugallta a gép —, hogy magyarázatot kapjak.

Nem akartam mások adatait bántani, ezért először csak lekérdezőprogram végrehajtására merészkedtem. A gép kérdezte: akarok-e programot végrehajtani. Mondtam, hogy igen. Jelezte, hogy ehhez a program kódjára-nevére lenne szükség. Ismerem? Mondtam, hogy nem. Erre kiírta, hogy mire képes.

Innen minden egyszerűen ment. A havi készlet átlagértékéről érdeklődtem, és a gép elmondta, hogy milyen adat alapján képes válaszolni. Kódokat is alkalmazott, de alul mindig mutatta, hogy melyik esetben mit kell írnom. Ha eltoltam, ismét jelezte: abba a dobozba azt kellett volna beütnöm, hogy ... Majd közölte az eredményt. A változtatást nem engedte meg: sajnos, nem ismer, ilyesmire én nem vagyok jogosult.

Majd leszek. Még egy-két órát játszottam ott, de jelentkezek a főnöknél, hogy ez a rendszer nekem kell. Kell, érted? Ajánlom a vállalatod figyelmébe is. Addig is szeretettel ölellek.

Váltsuk ismét komolyabbra a szót!

Essen szó most a fejlesztők felelősségéről! Valóban olyanoknak kell lenniük a számítógépes rendszereknek, amilyenek?

A magyar fejlesztő könnyen elhitetheti az addig teljesen tapasztalatlan felhasználóval és a régi gépes környezetben felnőttel egyaránt, hogy bizonyos megoldásoknak úgy kell lenniük. A fele sem igaz ... Ha valóban komoly gondról van szó, akkor pedig a fejlesztő azért felelős, mert nem közölte a végső felhasználóval: ez a rendszer pedig NEM mikrogépre való! Nézzük csak meg, hogy mitől jó vagy rossz egy mikrogépes rendszer?

A rossz fejlesztő összetéveszti a végső felhasználót magával. Egyik rendszeremnél szememre vetették, hogy a következő lapot miért K betűvel, a dokumentációt miért D betűvel lehet kérni a „megszokott" funkcióbillentyűk helyett. Megszokott? Kinek? Ha az én rendszeremben a betűt elütik, a képernyő nem változik lényegesen. Marad rajta — mondjuk — a K és a D. Ha egy funkcióbillentyűt elütök, akkor minden változik. Mi a kényelmesebb? Talán legjobb megoldás a két alternatíva együttes használata.

Rossz azonban a rendszer, ha a felhasználót a fejlesztő képére kényszeríti. Rossz a rendszer, ha a felhasználónak meg kell tanulnia olyan számítógépes funkciókat, amelyek a tudta nélkül kellene, hogy jól működjenek. Rossz a rendszer, ha a felhasználóra bízza a diszkettek váltogatását és összetett módon a kimentési-betöltési feladatokat. Aki autót vezet, nem feltétlenül akarja azt maga zsírozni, pucolni, talán még a parkolóházi szűk helyen leállítani sem. A gépkezelés olyan egyszerű, mint az autóvezetés. Ettől még nem szabad másodlagos feladatokra is kényszerítőnek lennie.

Valljuk be, hogy kevéssé tudunk online rendszereket tervezni. Fantáziánk csak az ikszedik — másnak! — tervezett rendszer után kezd kibontakozni. Nem szégyellendő tapasztalatlanságunk elsősorban a felhasználó-gép kommunikáció tervezésében muntatkozik meg. (Itt nem szabad feledninünk a korlátokat sem, sokszor 132-es, 160-as nyomtató helyett 80 pozicióra kell szorítkoznunk.)

Kellemetlen megoldás, ha a végső felhasználónak újra csak formanyomtatványokat kell előre kitöltenie — szerencsére ilyen torz rendszer egyre kevesebb látható. Ezzel szemben igen kényelmes, ha a felhasználó a képernyőn megszokott formátumú, elrendezésű bizonylatait látja viszont — akár a „természetesnek" megfelelő színezésben. Egy jól tervezett hagyományos bizonylat is tartalmazza — hátul, esetleg alul — a kitöltési utasítást. Egyre több bizonylatot látunk, amelyen a megfelelő rovat mellett — ha az kódolt — felsorolják a lehetséges értékeket is.

A többnyire elérhető 24 × 80-as képernyő persze kicsi. Távol esik a bizonylatszabványoktól, sajnos, még őrizve a lyukkártyák emlékét. Ha lassít is rendszerünkön, célszerű vagy lapozási lehetőséget adni a felhasználónak a kitöltési utasításhoz, vagy több képernyőből összetenni egy „bizonylatot", rajta a szükséges tudnivalóval. Ne kelljen a felhasználónak emlékeznie — az veszélyes és a hibák miatt időrabló —, és ne kelljen fél kézzel doksit forgatnia. Általában, különösen az azonosítók tekintetében, online rendszernél kicsit jobban takarékoskodhatunk a kódok alkalmazásával. Egyéb jellegű, osztályozó kódoknál ennek az igénynek ellentmondhat a szűk méret, és éppenhogy sarkallhat a kódok alkalmazására.

A felhasználó megszokta, hogy számára jelentős számú esetben az a bizonyos bemenet és kimenet azonos. Magyarul: egy papíron dolgozik. Kedvező tehát, ha a nem számítási, összegzési feladatoknál, az adatbeviteleknél és változtatásoknál a kimeneti kép a bemenetivel azonos. A felhasználó szeme önkéntelenül is a „szokott helyén" keresi az adatot. Adjuk meg neki ezt a minimális lehetőséget!

A jó tervezőnek arra is gondolnia kell, hogy a felhasználó időérzéke roppantul módosul kisgép — online rendszer — használata esetén. Tegnap még kitöltötte a bizonylatot, azt lyukasztották, hibaellenőrzés következett stb. stb. X nap, jó esetben óra múlva a felhasználó kapott kedvező vagy kedvezőtlen választ.

A gép előtt ülve ez az idő összezsugorodik. Az ember bámul a villódzó képernyőbe, és x másodpercen túl rágja az asztal szélét: no, mi lesz már? Az okos rendszer előre közli a felhasználóval, ha hosszabb a válaszidő: most felteheted a kávét, vagy ejtsd meg az esedékes telefonokat — inthet barátunk. Közepes türelmet igénylő válaszidő esetén (ez 5 másodperctől 15—20 másodpercig terjedhet úgy, hogy a felső zóna már a türelemhatáron van) a rendszer időnként közölheti, hogy most ott tart ...

A „fejreállt" rendszer

Becsapás ez. A felhasználó azt hiszi, hogy a gép gyorsan dolgozik. Egy nagygépen ugyanennyi idő alatt tízszer annyit végzett volna. Mi tagadás, még járnunk kell az online rendszerek pszichológiai iskolájába.

Minderről még sokat lehetne mondani. Egyetlen lényeges momentumba foglalható össze a tervezés alapproblémája. RÉGEN élesen szétvált a számítógépes feldolgozás és a végső felhasználó megszokott tevékenysége. RÉGEN a számítógépesek közutálatának tárgya volt az úgy lenézett ügyvitel. Hogyan és mit csinál a végső felhasználó? Kit érdekel! Adja csak az adatokat úgy, ahogy „a gép megköveteli!" MA ez a képlet megfordult. MA illenék úgy megtervezni azt a rendszert, hogy az tökéletesen beilleszkedjen a végső felhasználó rutinos munkavégzésébe.

RÉGEN a számítógép felhasználója akkor volt boldog, ha némi nehézségek árán megkapta az igényelt kimutatásokat, elemzéseket, adatsorokat. MA, kis barátunk mellett akkor fogja jól érezni magát, ha egyszerű kezelési módot nyújtunk számára — a gépen valamit előkeresni talán könnyebb is, mint a megfelelő dossziét megtalálni —, és életét a minimális mértékben változtatjuk. Ez van.

Meg még egy fontos dolog: fantázia az alkalmazási területek felfedezésében, amik mikrókkal támogathatók. Amelyek „gépesítésére" akkoriban nem is gondoltunk, mert ugye a forduló, a kényelmetlenségek ... E téren nincs is gond a magyar fejlesztőkkel. Inkább csak a buzdító, értő odafigyeléssel: már ne csak én sugalljam, drága végső felhasználó, hogy mit is lehet. Persze, vállalati érdekeltségre is szükség lenne ahhoz, hogy annyira fantáziadús legyen a jól jutalmazott tervező, mint otthon.

Valahogy így. Felhívásnál beadom: SZÜLINAP. A gép megkérdezi: legközelebbi vagy a múlt? A múltat választom. Kié? — kérdezi a gép. Mondom: NEJ. Erre kiírja, hogy három éve, tavalyelőtt és tavaly nejem egy csomag női cigarettát és egy blúzt kapott szülinapjára. Holnap valami mást kell vennem ...

Most fejlesztek

Marika? Nem emlékszel, hogy milyen kódok vihetők abba a mezőbe? Jé, ez a programrész ugyanaz, mint amott. Másoljuk át! Legalább egyforma lesz. Mikor is változtattam utoljára ezt a programot? Két hete?

Kis barátunk hallatlan előnye a rugalmas fejlesztési lehetőség. Ha kellőképpen felkészítjük, akkor már a feladat elején rögzíthetjük adott ismereteinket, fokozatosan bővítve és módosítva azokat igény szerint. Felállíthatjuk saját „mintaállományainkat", és a program azonnal tesztelhető. Hány „fordulót" kellett várni erre a hagyomá-

Vezessek talán naplót?

nyos magyar nagygépes környezetben? (Itt se tévedjünk el! A nagygépes fejlesztőrendszerekhez képest a mikrók csak nekünk jelentenek üdítő és nagyszerű lehetőséget!)

Nem akarok minderről sokat lamentálni, mivel a végső felhasználó — elsőre! — úgy érezheti, hogy ez neki nem fontos. A kicsit is tapasztalt fejlesztő pedig mindezt sokkal jobban tudja nálam. Tény, hogy munkatársam kipirultan mondta: Te, ezt a programot három hétig fordítottam volna a nagy IBM-en. Most két délután alatt kész lett!

Kicsit el is kényezteti az embert ez a készenlét. Magamon tapasztaltam, hogy hajlamos vagyok a lazaságra, a kevésbé szigorú átgondolásra. No, bumm, ha nem megy, akkor azonnal javíthatok. Régen ezért a fordulószámokban győztesek szégyentáblájára kerültem volna egy komoly cégnél.

Ne tévessze meg sem a fejlesztőt, sem a végső felhasználót ez a kétségtelenül rugalmas fejlesztési lehetőség! Nehéz vele nem visszaélni. A gép villogva figyelmeztet, hogy válaszra vár: hát mondok neki valamit. Régen, a ceruzát rágva kicsit jobban megfontoltam a választ. Mindehhez sok köze van a végső felhasználónak is. Talán nem a várt gyorsasággal készül el a rendszere és talán kicsit tetűlassú lesz. Mert minden alkalmazásfejlesztési készség, prototípus-alkotási lehetőség ellenére kis barátunk nem kényszerít a JÓ tervezésre, az OPTIMÁLIS terv kialakítására. Sőt! Másokat úgy sem zavarunk, majd csak elketyeg a masina alapon lazábbnak érezzük a hatékonysági követelményeket.

Kétpályás játék

A régi, nagygépes környezethez hasonlóan kezdtem fejleszteni új rendszeremet. Nem, én nem vagyok programozó, „csak" rendszerszervező. Szóval diktáltam — jól-rosszul — a feladatot a programozónak, aki kellemetlenül visszakérdezett ostobaságaimra. Majd egy kis irkálás a papíron, és ketyegtette a gépet.

Ha már van, megpróbálom — gondoltam. Tanultam valamikor programozni — 1971-ben. Vagyis nem tudok. Itt valóban egy magas szintű nyelvvel dolgozunk, amely roppant hasonlít az angolra. Nem kell megijedni, negyven-ötven szóval megél az ember. És utána minden mennyire más!

Tudom, hogy nevetséges, de nekem kell kiszámolnom, hogy mennyivel kevesebb ebédjegy és kilométerpénz jár nekem, ha szabadságon, kiküldetésen stb. vagyok. Elvégre a kétszázezer adminisztrátort nem ezért tartja vállalatunk! Ez van. De nem fogok én mindig plajbászt fogni, mert minek. Nekiveselkedtem, és írtam egy tízsoros programot.

Most pedig angolra kell fordítanom a rendszer dokumentációját. Eddig Zsó gépelte, miután én szintén legépeltem. Ő adott neki külcsínt. Amíg volt türelme. Mert most itt ezt a bekezdést cseréld ki, amott hagyd ki azt a részt, itt pedig szúrjuk be — változtattam a véleményemet. Valamint az oldalszámot, a tartalomjegyzéket, a hivatkozásokat. Most, most megtanultam a „funkcióbillentyűket" és a szövegszerkesztő használatát. Bár nem használ a szememnek, direkt fordítom az anyagot a gépbe. A végén majd — talán —, szépen legépeli Zsóka.

Nem folytatom. Csak annyit akarok mondani, hogy valóban barátunk ez a számítógép. Ha olyan a végső felhasználó, akkor végtelenül barát rendszert kell készíteni a számára. Erre van mód; ez a végső tanulság a fejlesztőknek. A felhasználó pedig érzékelhetné, hogy kettős értelemben is megnyílt előtte a pálya. Ha kicsit beleásná magát — nem nagy fáradsággal! — kis barátunk lelkivilágába, akkor többet és értelmesebben követelhetne a fejlesztőktől. Ez éppen úgy a saját érdeke, mint a saját játék most nem kell fejlesztő, most ez megcsinálom magamnak. Nem ne héz.

A búcsú

Sokat szövegeltem eddig barátunkról, a számítógépről. Ezt nem is lehet befejezni, csak abbahagyni. Adósságot is leróttam: a mikrogép mégsem aerobic.

Zárásképpen szeretném összegezni néhány „üzenetemet", ha nem hangzik nagyképűnek ez a kifejezés. Nézzék el nekem, mert a jó szándék vezetett. És a ritka helyzet. Fejlesztő is vagyok, végső felhasználó is. Értek is a számítógépekhez, meg — olyan igazán — nem is. Meg tudnék fejteni egy magas szintű programot, de nem mindegyiket tudnám jól megírni. Szóval picit a két véglet között állok, és célom e kettő érdekeinek és lehetőségeinek egyesítése. Ezért írtam.

Remélem, hogy az ifjú tehetségek egy picit már látják, hogy a mikrogép nem pusztán játék, nemcsak szórakozás. Ugyanakkor — ha netán nem szeretett kötelező tárgyról lenne szó (remélem, hogy nem) —, a programozás nemcsak kín. Tehetek, amit akarok, ameddig magam vagyok a fejlesztő és a felhasználó. Ha azonban hivatásérzetből végzem ezt a vonzó munkát, akkor el kell feledném fejlesztő mivoltomat, és át kell kapcsolnom a végső felhasználó hullámhosszára. Nem a BASIC nyelv a megváltó; nem ennek tudása a végcél. A géptől, a nyelvtől, a számítástechnikától nem tanulok feladatmegoldó készséget. Legfeljebb adott, szűk struktúrákban segít eligazodni a gép. Ha nem így lenne, akkor ugyebár, csak jó rendszerek születnének. Teszik? Ha majd ifjú kollégám, ugyanúgy képes leszel figyelni a raktáros, a főkönyvelő, a személyzeti osztályvezető igényeire, mint a gép követelményeire, ha majd az ő szavukat ugyanolyan pontos kinnal kell betartanod — és ezt örömmel, a cél érdekében teszed —, mint a DO CASE—END-CASE párost, akkor megértél: szakemberré váltál. Addig csak amatőr vagy.

A kevésbé fiatal korosztály végre megpróbálhat kilépni eddigi szűk kereteiből. Kicsi odafigyeléssel naponta tudna érvelni. Nem igaz, hogy a gép azt úgy követeli. Nem valós, hogy amit kívánok, az lehetetlen. Téged csak akkor fizetlek meg, fejlesztő, ha valóban ízlésemnek megfelelően alakítod a rendszert. Ehhez azonban az kell, hogy ez a korosztály megismerkedjen a gép közvetlen természetével. Netán még az általam nem mindig helyeselt funkcióbillentyűkkel is. Talán a magasabb szintű nyelvek általános képességeivel és legvégül magával a nyelvvel is. Ez a könnyebb feladat. Megváltozni: ez a legnehezebb. Mert, ne feledjük, a fejlesztő is értékelni a felhasználót!

Nálunk számítógépes bérszámfejtés van, de nekem kell megadnom a kajajegyek mennyiségét: mikor voltam szabadságon. Nos, ott van a szabadságpapíron. Miért nem vonja le a gép? Talán a felhasználó volt tehetetlen a rendszer kijelölésekor? Talán nincs fantáziája, hogy az általam korábban említett feldolgozási láncban megtalálja a helyes arányokat? Talán a fiatalok mégsem tanulhatnak tőlünk feladatkijelölést és megoldási készséget? Nagy kópé ez a mikró: hamar hű képet ad rólunk, alkalmazókról!

A mikró hamar visszarúg, szemben a lehetetlen és értelmetlen feladatokat is zavartalanul — bár drágán —, emésztető nagygépekkel. Hamar kiderül, hogy az információs rendszer működtetésének költségei kettősek: emberi és technikai költségtényezőkből tevődnek össze. Nagygépnél főleg az utóbbi volt drága. Most, látszólag, olcsó. Mennyi emberi ráfordítás árán? Könnyen fejlesztünk alkalmazásokat, de vigyázat! Sokat vagy kidobunk, vagy rabszolgájává válunk. Előny: legalább nem akarunk többpéldányos, több száz oldalas tablókat készíteni, amire a mikró nem való. Testre szabottabb lehet az információ. Hátrány: a mikró egy ideig tűri, hogy vele imitáljuk a nagygépet. Majd váratlanul áll fejre, mármint az a rendszer, amit csak nagygépre lett volna szabad tervezni.

Végül: újra kell együtt tanulnunk ezt a szakmát fejlesztőknek és végső felhasználóknak. Ami a nagygépek esetén kívánatos, de lehetetlen volt, az most elérkezett. Többet kell tudnia a sikerre vágyó fejlesztőnek a felhasználói mindennapokból. És mód van arra, hogy többet ismerjen a jó rendszert kívánó végső felhasználó a számítástechnika érintkezési felületéből. Ez barátunk legfőbb érdeme: szűkül a szakadék.

Most pedig nézem tizenhárom éves lányomat, aki bekapcsolta a kis IBM PC-t és akinek — egyelőre mint „végső felhasználónak" —, szeretettel magyarázom, mit hogyan. Legelső alkalom; még nem sokat ért mindebből. Mégis megdöbben, hogy pillanat alatt lekérdezheti társasági programjait, mert már vannak ilyenek. Öt percet bibelődünk a géppel, felárazva magántárgyait, és máris meglepődik azon, hogy mennyi mindene van. Gyerekekre szabott készletnyilvántartás.

Talán majd számítástechnikus lesz, talán nem. Szeretni fogja ezt a kis masinát. Előbb még hosszan kell nevelnem, hogy értse meg a mások igényeit. Hogy tudja pontosan megfogalmazni saját feladatát. Ehhez már csak kiegészítés az a csöppnyi technikai tudás, amit barátunk megkövetel.

DR. HALASSY BÉLA

FÓRUM

Rendellenességek és rendetlenkedések

Az 1986. szeptemberi számban a 8. oldalon leírt lemezoldal-jelölés „ügyes" megoldás, és a legjobb módszer arra, hogy a lemez állományait tönkretegyük. A lemezen levő ID ugyanis nem a felhasználók tájékoztatása céljából van a lemezen, hanem azért, hogy a lemezegység operációs rendszere értesülhessen a lemezcseréről. De ha két különböző lemezen, vagy ugyanannak a lemeznek a két oldalán ugyanazt az azonosítót találja, nem veszi észre a cserét, és a korábban használt lemez foglaltsági térképe (BAM-ja) alapján ír a mostani lemezre is, esetleg felülírva programjainkat, de mindenképpen felülírva az új lemezoldal BAM-ját. Ne sajnáljuk tehát a fáradságot, és jelöljük a különböző lemezoldalakat különböző ID-vel!

Tetszett a 29. oldalon olvasható „Egy új BASIC utasítás", az INPUT+. A középső hasában található assembly listába azonban hiba csúszott. A 270-es programsorban a műveleti kód nem STA, hanem STY. A 2. listán levő BASIC betöltőprogramban már a hibátlan kód szerepel.

Értetlenül olvastam a 29. oldalon kezdődő „GET* helyett INPUT*" című cikket. A problémának egy sokkal egyszerűbb és jobb megoldása található „A VC-1541-es lemezegység programozása" című könyv 251. oldalán kezdődő fejezetében. Az INPUT* rutinnal szinte korlátozás nélkül, maximálisan 254 karakter hosszúságú rekordok olvashatók be a lemezről. A beolvasott rekord tartalmazhat vesszőt, kettőspontot, pontosvesszőt, sőt 13-as ASCII értékű karaktereket is. Egyedül a CHR$(0) karakterek olvasása okozott — a feladat szempontjából áthidalható — problémát.

Ki ad magyarázatot a 38. oldalon olvasható „Ki ad magyarázatot?" című írással kapcsolatban arra a problémámra, hogy 1400 ill. 1900 hogyan lehet 300-1000 közötti szám? Lehet, hogy csak egy jel hiányzik valahonnan? Talán 10 000 volt a felső határ? (Igen, sajtóhiba volt; a helyes intervallum 300-10 000. — A szerk.)

BARNA LÁSZLÓ

A μM szerkesztői nagyon sok levélre személyes választ küldenek, ha a probléma nem bírja el a több hónapos válaszadást. Itt most megpróbáltunk néhány közérdekű levelet összegyűjteni és ezekre válaszolni.

Esztergár Pál, Budapest,

Fehérvári út 35. 1117

Eleget téve a — néhány hónappal ezelőtti — felszólításnak, ezúton keresem a MIKROSZÁMÍTÓGÉP MAGAZIN olvasói között a CPC (Schneider, Amstrad) számítógépek tulajdonosait. Én ugyanis ezzel a — Nyugat-Európában az egyik legelterjedtebb és nagyon sokat tudó — géppel, továbbá hozzá elég bőséges irodalommal, programokkal rendelkezem, és mivel jól állok tapasztalatok dolgában, szívesen adok tanácsot mind a „LOCOMOTIVE BASIC", mind a gépi nyelv vonatkozásában, de én is örömmel venném mások tapasztalatait.

Úgy hiszem, hogy az a kérésem, hogy a MIKROSZÁMÍTÓGÉP MAGAZIN tegyen közzé egy — CPC tulajdonosokat kereső — felhívást és regisztrálja a jelentkezőket, nem ellentétes a lap célkitűzéseivel, és elősegítheti ezen tényleg sokat tudó és sokoldalú géptípus néhány hazai példányának célszerű hasznosítását.

A turistaforgalom növekedésével egyre színesebb lesz az itthoni házi-számítógép tábor. Ezért adunk teret szívesen a hasonló felhívásoknak; találjanak egymásra az azonos gépeket használók! Szeretném ezzel a μ Klub mozgalmat is egy kissé felélénkíteni.

Menyhárt Tibor, Debrecen,

Gyepsor u. 12. 4031

A Mikroszámítógép Magazinban és egyebütt közölt cikkek felkeltették az érdeklődésemet a FORTH programnyelv iránt. Bár sikerült hozzájutnom a ZXFORTH 1.1 programhoz, a ZX81-re írt ZXFORTH leírásához sajnos a mai napig sem.

Nem lehetne levelemnek ezt a részét „AZ OLVASÓ ÍRJA" rovatban lehozni? Talán valamelyik olvasó tudna nekem segíteni. Cserébe COMPILER-t, ASSEMBLER-t, DISASSEMBLER-t, TOOLKIT-ot, GRAPHICS TOOLKIT-ot, valamint játékprogramokat tudnék felajánlani.

Közöltük.

Gerlits Péter, Budapest,

II., Kelemen L. u. 2. 10. ép. I. 2. 1026

Segítségüket szeretném kérni a lapjuk ez év júniusi számában megjelent „Ékezetes ábécé C64-re" programmal kapcsolatban.

A programot többszöri próbálkozás ellenére sem sikerült megvalósítani. Rögtön az elején a POKE 44,16: POKE 45,61: POKE 4096 + 64,0 NEW parancs után a gépem syntax hibát jelez. Ez utasítássor nélkül legépelt program pedig indítás után befagy. Lehet, hogy bennem van a hiba, de ha esetleg a sajtóhiba ördöge incselkedett, kérem, írják meg a helyes megoldással együtt, mert nagyon szeretném, ha a gépem végre tudna magyarul.

A program hibáinak javítását a μ M októberi számában közöltük.

Balogh Gábor, Sopron,

Madách u. 8. 9400

A szeptemberi Mikroszámítógép Magazinban ovastuk Pivarnyik Attila javaslatát, amellyel teljesen egyetértünk.

Ezzel a levéllel jelezni szeretnénk jelentkezésünket annak a két oldalnak a szerkesztésére — szerkesztésének segítése — amely a SPECTRUM számítógéppel foglalkozna.

Amennyiben igényt tartanak segítségünkre, kérjük jelezzék a fenti címen.

Ötleteinkből néhány:

— a gép néhány trükkje;
— rövid felhasználói programok (BASIC, gépi kód);
— rövid játékprogramok (BASIC, gépi kód);
— programozástechnikai problémák;
— játékrovat: hír, riportok, egyéb tanácsok;
— hosszabb gépi kódú vagy BASIC programok közlése több számon keresztül;
— SPECTRUM TOP lista (játékprogramok).

Volt egy régi szerkesztőségi cikkem, amelynek az volt a címe — ha jól emlékszem — hogy „Nyitott Szerkesztőség", ami röviden azt jelenti, hogy szeretnénk a számítástechnika iránt érdeklődőket, főleg ha íráskészséggel is rendelkeznek, a szerkesztőségi munkába bekapcsolni. Nagyon sok ötletet kapunk, de sokkal kevesebb írást. Csak az elkészült cikkből tudjuk megítélni, hogy vállalkozunk-e hosszú távú együttműködésre vagy nem vállalkozunk, mert esetleg se a beküldött írás tartalma, se a stílusa nem üti meg azt a mértéket, amelyet munkatársainktól elvárunk.

Külső munkatársakkal az együttműködés úgy szokott elindulni, hogy küldenek egy írást, az megjelenik, azután jön a következő. Később az illetők eljárnak a szerkesztőségi megbeszélésekre, és észre sem vesszük, máris szoros a kapcsolat, számítunk egymásra. Sajnos, ennek az ellenkezője fordul elő legtöbbször. Jön egy ötlet, esetleg egy írás, utána pedig hiába várjuk a folytatást; rendszeresen írni egy havonta megjelenő magazinba nem kis vállalkozás.

Várjuk az írást másoktól is.

Fekete Sándor, Oradea 3700, Bihor, Romania

Breiner Béla u. 59/A

Le kell szögeznem, hogy én már többször írtam önöknek. Tehát röviden rátérek a témára:

Nagyon szépen kérem önöket, ha lehet, adják közre a levelező rovatban e pár következő sort; persze lehet átdolgozva is, úgy, hogy a lényeg megmaradjon. Szeretném, ha közös erővel és elég kitartással létre lehetne hozni egy univerzális jellegű modern jelvényt, amely összekapcsolná a számítástechnikát szerető amatőröket; tehát ez valahogyan segítene abban, hogy a rokonlelkek egymásra találjanak. Nagy álmom, vagyis vágyam, hogy ez ne csak Magyarországon legyen érvényes, hanem ha nem is az egész világon, de Európában; sőt lejjebb is adom: vezessük be Európának ezen a részén. Lehetne jelvény helyett érme, esetleg plakett stb., stb. Van valakinek véleménye, hát az ötletemhez hozzáfűzhet valami megjegyzést, észrevételt stb. stb.

Az igazság az, hogy én még nem terveztem meg a jelvény formáját és tartalmát, de ha gondolják, én megcsinálom.

Az ötlet, amelyet hűséges levelezőnk küldött Erdélyből, érdekes. Azt hiszem, hogy nem vagyunk jelvény ügyben kompetensek, így nem merem biztatni, hogy kezdje el a tervezést. A μ kluboknak van már egy jelvényük, talán alkalmas lehetne a szoftver és a hardver amatőrök összetartozásának kifejezésére is.

Török György, Gomba,

Rákóczi út 24. 2217

Önök a Mikroszámítógép Magazin 1986. júliusi számában közöltek egy cikket a fényceruza készítéséről ZX-SPECTRUM-hoz. Szeretném, ha közölnék, hogy hogyan lehetne fényceruzát készíteni C—64-hez.

Talán egyik olvasónk tudna segíteni.

Várjuk a cikkét.

Juhász Sándor, Csánytelek,

Pusztaszeri út 59. 6647

Örömmel olvasom az Önök lapját, mert minden számban van valami, ami érdekel. A közeljövőben én is szeretnék egy számítógépet. (Főiskolai hallgató vagyok a kecskeméti Gépipari és Automatizálási Műszaki Főiskolán, szüleim pedig kisiparosok.) A belföldi árak azonban igen magasak. Nemrégiben olvastam az Önök lapjában, hogy az ATARI 800XL számítógép, hozzá nyomtató plusz floppy, olcsó monitor és egy szövegszerkesztő program 1140 DM. Ezzel kapcsolatban lenne néhány kérdésem.

1. Bécsben mennyivel térnek el az árak az NSZK-tól (általában)?
2. Nem csaléka-e az alacsony ár, nem megy-e a gép rovására?
3. Önök szerint érdemes-e a fenti konfigurációt behozni, vagy valamilyen más típust javasolnak?

Más. Mi a véleményük a TV Computerről és a PRIMO-ról? Esetleg ezek valamelyikét vásároljam? Ezeken milyen a hang és a grafika?

A levélből csak azokat a részleteket közöltük, amelyekre „időtálló" választ tudunk adni.

Elég sok hasonló levelet kapunk, amelyekre részben levélben válaszolunk, de sokszor ebben a rovatban is.

Általában az NSZK a „legolcsóbb" európai ország, ha valaki számítógépet, de általában elektronikus berendezést akar vásárolni. Ez nem zárja ki azokat az eseteket, amikor egy alkalmi kiárusítás alkalmával pl. Bécs is lehet nagyon olcsó. Az 1140 DM, ha akarom alacsony, ha akarom, magas ár. A nagyon alacsony ár mögött nagyon sokszor távol-keleti szállítót találunk, ilyenkor nem ritka a minőségi probléma, amelyre néhány olvasói levélben is találunk példát. Ami a típust illeti, én mindig olyan gépet vennék — ezt többször megírtam már —, amiből sok van az országban, és amihez szervizt is tudok találni. Legjobb tudomásom szerint ennek a követelménynek a külföldi gépek közül ma csak a Commodore felel meg.

Ami a hazai gépeket illeti, én pártolom a magyar ipart. Nem megvetendő előny, ha az ember a problémáit itthon el tudja intézni. A hazai választékból már csak a TV Computer maradt meg, a többi gyártó az olcsó gépekkel nem bírván a versenyt, „bedobta a törülközőt".

Papp László, József A. Fiúkollégium,

Székesfehérvár, Széchenyi u. 13. 8000

Én egy fehérvári diák vagyok, s szabad időmben programokat írok C—64-re. Írtam egy programot, egy új SAVE rutint, amelynek segítségével tetszőleges memóriarészt lehet elmenteni. Az Önök lapjában szeretném közzétenni.

Ha ez lehetséges, kérem írják meg, milyen leírást vagy listát kell küldenem a programról, és milyen feltételek mellett közlik.

Csak kipróbált és nyomtatón listázott programokat fogadunk el. Egy sorban 40 karakternél több ne legyen, különben a listát nem tudjuk jól tördelni. A szerzők küldjenek jó programleírást. Ne feledjék, nem a padszomszédjuknak készül az írás, hanem egy újságnak, így legyen részletes, érthető, de ne legyen fecsegő, mert azt senki sem szereti.

Somfai Gábor, Budapest,

Rákóczi út 8/B. 1072

Lapjuk régi olvasójaként az alábbiakban összegyűjtöttem néhány olyan — technikai, illetve koncepcionális — ötletet, amelyek talán segíthetnek abban, hogy a lap színvonala még jobb legyen.

1. Kezdjük mindjárt a tartalomjegyzékkel. Sajnos a lapban szereplő legérdekesebb cikkek egy része a tartalomjegyzékben eleve nem szerepel. (Például a 86. júliusi számban a tartalomjegyzékből kimaradt a „Mibe kerül a 16 bit" című cikk, amely szerintem ebben a számban a legérdekesebb írás volt. (Javasolom azt is, hogy a tartalomjegyzék a cimek mellett minden egyes esetben tüntesse fel azt is, hogy milyen géptípusra vonatkozik a cikk, ha ez lehetséges.) A lap jelenleg annyira eklektikus ugyanis, hogy nem ártana, ha már az első szemrevételezésnél mindenki látná, hogy egyáltalán foglalkoznak-e kedvenc gépével az adott számban.

2. A lapban közölt programoknál sok esetben teljesen olvashatatatlanok a grafikus jelek. A legtöbb nyugati számítástechnikai lap ezen úgy segít — bár nyomdatechnikailag erre kevésbé vannak rászorulva —, hogy nem a képernyőn megjelenő jelet ábrázolják, hanem azt, hogy melyik billentyűt kell lenyomni a létrehozásához, ez általában jobban kisilabilzálható. Arra lenne szükség, hogy a szöveg közben ne közöljenek programokat, ezeket a lap végén közöljék, géptípusonként csoportosítva, rövid magyarázattal arról, hogy a programot hogyan kell begépelni. Van egy másik segítség is: a jobb lapok ma már úgynevezett „proofreaderekkel" közlik a programokat, a profreader program betöltése után az új program minden egyes soránál valamilyen betű vagy számkombináció jelzi, hogy az adott sort jól gépeltük-e be, nem a program végén derül ki, hogy az ötödik sorban hiba van. Javaslom, hogy a hibajavításokat mindig a programrész végén közöljék, jelenleg ugyanis teljesen véletlenszerűen bukkanok rá korábbi számokban megjelent programok javítására.

3. Magyarországon ma a felhasználók nagy része csere útján jut a programjaihoz, illetve az azokra vonatkozó irodalomhoz. Közöljenek rendszeres összeállítást a hazai klubhálózatról, bemutatva a klubokat, címmel, az ott preferált géptípusok feltüntetésével stb., ez jelentősen megkönnyíthetné az ismeretek bővítését és az ehhez szükséges kapcsolatok létrehozását.

4. Fordítsanak nagyobb figyelmet a laikus felhasználókra, azokra, akik a gép működését, a gépi kódú programozást stb. még nem ismerik, illetve sohasem fogják megismerni, mégis használni akarják gépüket anélkül, hogy számítástechnikai szakemberekké válnának. Szerintem a géptulajdonosok közül sokan tartoznak ehhez a körhöz. Néhány tipp:

— Segítsenek eligazodni a géptípusok és programok tengerében. Olvasóik nagy része véletlenszerűen jut programokhoz, nincs áttekintésük arról, hogy milyen a kínálat, egy-egy célt milyen program szolgálna legjobban. Példamutatónak tartom e tekintetben a C—64 „Software alkalmazói segédlet" III. kötetében közölt összeállítást, amely hét adatfeldolgozó programot hasonlít össze, megkönnyítve az eligazodást. Javaslom, hogy a főbb géptípusokra készítsenek hasonló összefoglaló értékelést, legalább a szövegszerkesztő, a spreadsheet és a másoló programokról.

— A nem szakértő felhasználó sokszor tengernyi irodalom átrágása után sem képes tisztába jönni a legelemibb trükkökkel, amelyek a gépe működtetéséhez szükségesek. Két példa:

1. A programok betöltése. Nem lenne érdektelen áttekinteni a betöltés fortélyait, azt a 10-20 trükköt, amellyel egy laikus rájöhet arra, hogy a kezében tartott mágneslemezről betöltött program miért nem indul el (pl. a programhoz bootot kell keresni, a program egy másik program része, gépi kódban íródott, és ezért más utasítással kell betölteni, nem a run parancs indítja, hanem valami más, pl. sys utasítás stb.). A géptulajdonosok nagy része ilyen elemi problémákkal küszködik.

2. Özöne létezik a másolóprogramoknak. Érdemes lenne összegezni azt, hogy mikor melyiket érdemes használni, hogyan kell lemásolni a lemásolhatatlan programokat. Saját példám alapján írom mindezt, én például heteket töltöttem azzal, hogy megpróbáltam lemásolni a „Master 64"-et, míg végül rátaláltam a B 52 című másolóprogramra, amellyel ez sikerült, és amelynek a létezéséről sejtelmem sem volt.

5. Közöljenek több ismertetést az egyes gépek alkalmazásáról, konkrétabban ismertetve a megoldandó problémákat, a felhasználók számára is érthetően. Így például egy adott adatrendezési szoftvernél engem nem túlzottan *érdekel*, hogy a memóriában a program mit hova tett stb., érdekel viszont az, hogy a rendszer mekkora adathalmazt kezel, milyen típusú programmal stb. Mindezekből ugyanis rá tudok jönni arra, hogy a saját szakmai problémáimhoz milyen típusú megoldást kell keresnem, mielőtt még megtanulnék programozni. Így például adatrendezési problémáimnál én még nem tudom összehasonlítani a C—64-re készült „Superbase" és „Datamat" programok gépi kódját, számomra az a fontos információ, hogy a „Datamat" előnye a gyors, egyszerű kezelhetőség, de a „Superbase"-t kell használnom olyankor, ha bonyolultabb rendezési folyamatokat akarok elvégezni stb.

6. Sokkal több információt olvasnék szívesen a mikrogépek külföldi áráról, aktuális piaci helyzetükről. A hazai árakat botrányosnak tartom, minősíthetetlen az, hogy a monopolhelyzetben levő hazai gyártók valamiféle kreált belső árral hasonlítják össze saját áraikat, így igazolva a nyugati árszintet akár tízszeresen is meghaladó áraikat. Ilyen témában foglalkozhatnának a magánimport nehézségeivel, a vámokkal stb. is, elég sok mondanivaló lenne ezen a területen is.

Somfai Gábor levelénél jobb befejezést nem tudtam volna találni, tanácsait köszönjük, megszívleljük.

További jó együttműködést kíván a szerkesztő:

KOVÁCS GYŐZŐ

PC piaci prognózis

A nyugatnémet TEDAS adatbank — amely a Mikrocomputer Zeitschrift folyóiratkiadó vállalat on-line szolgáltatása — egyik legutóbbi összeállításában értékelte a személyi számítógépek piacát. E szerint a személyi számítógépek gyártói eddig 13 millió darab PC-t értékesítettek. Az elkelt berendezések száma — amennyiben a jelenlegi tendencia folytatódik — 1990-re el fogja érni a 40 millió darabot.

Európában 1985-ben mintegy 4,6 milliárd dollár értékben adtak el személyi számítógépeket. A jelenlegi piaci trendek még egy ideig várhatóan nem fognak lényegesen megváltozni, csak az arányok tolódnak el. 1992-re a becslések szerint a másodgépek, valamint az elavult berendezések pótlására megvásárolt új gépek a forgalom több mint ötven százalékát fogják kitenni.

1987-ben várhatóan 1,7 millió eredeti IBM PC és ezzel szemben mintegy 2 millió (!) IBM PC kópia — a Távol-Keletről és egyéb országokból származó hamisítvány — kerül a felhasználókhoz a téma szakértőinek mértékadó véleménye szerint.

kj

Az 1986. októberi szám 25. oldalának utolsó mondata szerint „az igen jónak bizonyuló gyors rendezőt . . . a szerencsések megtalálhatják a könyvben".

Olvasóinkat „szerencsésség" tesszük, és közöljük mind a gyors rendező BASIC-programját *(1. lista)*, mind a Shell—Metzner rendezőét *(2. lista*, mely egy utasítással rövidebb, mint a cikkben a könyvből idézetté) a legtöbb gépen futó BASIC-változatban. Mindkét programban, ha a tételek száma több mint 100, a 10. sort módosítani kell.

```
5 REM GYORSRENDEZOE
10 DIM A(100),F(15),L(15)
20 INPUT"TEETELSZAAM";N
21 FOR I=1 TO N
22 A(I)=1+INT(RND(0)*N):PRINT A(I);:NEXTI
30 S=0:F=1:L=N
40 M=A(INT((L+F)/2)):I=F:J=L
50 IF A(I)<M THEN I=I+1:GOTO50
60 IF A(J)>M THEN J=J-1:GOTO60
70 IF I>J THEN 120
80 IF I=J THEN 100
90 T=A(I):A(I)=A(J):A(J)=T
100 I=I+1:J=J-1
110 IF I<=J THEN 50
120 IF I>=L THEN 140
130 F(S)=I:L(S)=L:S=S+1
140 L=J:IF F<L THEN 40
150 IF S=0 THEN 170
160 S=S-1:F=F(S):L=L(S):GOTO40
170 FOR I=1 TO N:PRINT A(I);:NEXT I
```

```
5 REM SHELL-METZNER RENDEZOE
10 DIM A(100)
20 INPUT"TEETELSZAAM";N
21 FOR I=1 TO N
22 A(I)=1+INT(RND(0)*N):PRINT A(I);:NEXTI
30 M=N
40 M=INT(M/2):IF M=0 THEN 120
50 K=N-M:J=1
60 I=J
70 L=I+M:IF A(I)<=A(L) THEN 100
80 T=A(I):A(I)=A(L):A(L)=T:I=I-M
85 IFI>=1 THEN 70
100 J=J+1:IF J>K THEN 40
110 GOTO60
120 FORI=1 TO N:PRINT A(I);:NEXT I
```

Könyvek

Vadnai Szabolcs:
Commodore 64 programozói zsebkönyv
(Budapest, 1985.
Novotrade Rt. 68 oldal.
Ára: 149,— Ft.)

A zsebkönyv célja, hogy az olvasónak egy helyen összegyűjtve szolgáljon mindazokkal az információkkal, melyeket a C64 amatőr és szakember-felhasználói eddig csak fáradsággal, több könyvből tudtak összeszedni. A kötet főbb fejezetei: BASIC, Simon's BASIC, Hang és zene, Grafika, Gépi kódú programozás, Help kártya használata. A függelékben kódtáblázatokat és a mágneslemezekkel kapcsolatos tudnivalókat találja az olvasó.

1001 játék és a
GRAPHICS BASIC
Commodore 64-re
(Írták: Erdős Iván, Schmidt Endre, Németh István, Székely László)
(Budapest, 1985.
LSI ATSZ, 115 oldal.
Ára: 100,— Ft.)

A kötet gyerekeknek, felnőtteknek, mindazoknak szól, akik C64 számítógép tulajdonosok vagy kedvelők, akik szórakozni és játszani szeretnének a számítógéppel. Hazánkban igen sok számítógépprogram van forgalomban, melyek használatát nem mindenki ismeri. Ez a kötet tartalmazza a legelterjedtebb C64-re írt játékok ismertetését és egy 1000 címszóból álló minilexikont is, amely útmutatóul szolgál az ügyességi, logikai, stratégiai játékok közötti eligazodáshoz.

Azoknak a felhasználóknak, akik saját játékötleteiket kívánják megvalósítani, a könyv harmadik fejezete nyújt segítséget, melyben a GRAPHICS BASIC című BASIC-bővítőt ismertetik a szerzők. Ennek használatával igen egyszerűvé válik a C64 számítógép képernyőkezelése.

Hofmanné Boskovitz Éva:
A Z80 assembly programozása
(Budapest, 1985.
LSI ATSZ, 368 oldal.
Ára: 249,— Ft.)

A könyv azok számára készült, akik a Z80-as mikroprocesszorral még nem foglalkoztak, és akiknek az assembly programozás még nem a szakterületük.

A kiadvány főbb témakörei: a programfejlesztés rendszertechnikai eszközei, gépi kód, assembler, magas szintű nyelv, adatmozgatás, Z80 utasítások, programok, bemeneti-kimeneti műveletek, adatkezelés.

A példaprogramok tematikusan épülnek egymásra, és sok programozási fogást, trükköt tartalmaznak. A könyv ismerteti a Magyarországon leginkább elterjedt fordítóprogramok kezelését is.

Szűcs Pál:
Video kézikönyv
(Budapest, 1985.
OMIKK, 323 oldal.
Ára: 117,— Ft.)

A kézikönyv a hazai videofelhasználók számára foglalja össze azokat az információkat, melyek a videózás megkezdéséhez nyújtanak segítséget.

A kötet a televíziós technika alapjait, az NTSC, a SECAM és a PAL rendszert ismerteti. Leírja a videokamerák fajtáit, a képmagnetofonok, a monitorok, a tévé-vevőkészülékek és a videovetítők legfontosabb jellemzőit. Ismerteti a video fejlődésének új irányait: a műholdas és a kábeltelevíziót, a képelemezeket és a videotex rendszereket. Végül röviden összefoglalja a video használatát az oktatásban.

Bárdos Attila—
Körtvélyesi Gézáné:
Programozási alapfeladatok gyűjteménye
(Budapest, 1985.
SZÁMALK, 209 oldal.
Ára: 101,— Ft.)

A 444 programozási alapfeladatból álló gyűjtemény célja a programozás tanulásának és tanításának elősegítése. A kötet szerzői olyan fogásokat, módszereket ismertetnek, melyek begyakorlása révén jártasságot szerezhet az olvasó a problémamegoldás területén. A feladatok nehézségi foka az egy-két utasítástól a 2-300 utasításnál is több eljárást tartalmazó bonyolultabb programokig terjed.

A feladatgyűjtemény témái: elemi algoritmusok, ciklusok, alapalgoritmusok, adatfeldolgozási algoritmusok, vegyes feladatok. A fejezeteken belül a nehézségi fok szerint csoportosítja a feladatokat, melyek megoldásához számos ötletet kínál. Ezekhez az önálló megoldások helyességének ellenőrzését segítő programterv, algoritmus és program kapcsolódik.

Madarász László:
Digitális CMOS kapcsolásgyűjtemény
(Budapest, 1986.
Műszaki Könyvkiadó,
204 oldal. Ára: 93,— Ft.)

A könyv a CMOS technológiával készülő digitális integrált áramkörök felépítésével, típusaival és alkalmazásával foglalkozik. Közel száz áramköri részletmegoldást és több mint 80 komplett kapcsolást ismertet. A szerző közli a kapcsolásokban felhasznált integrált áramkörök bekötési rajzát és főbb adatait is, ez önmagában is nagy érdeklődésre tarthat számot. Mivel az eddig megjelent, digitális technikával foglalkozó könyvek a CMOS áramkörök gyakorlati alkalmazásával csak érintőlegesen foglalkoztak, ez a könyv mind az ipari felhasználók, mind pedig az amatőrök számára a legújabb ismereteket nyújtja.

Mi micsoda magyarul a számítástechnikában?
Mikroszámítógépes értelmező szótár
Szerkesztette: Kis Ádám
(Budapest, 1986.
Tömegkommunikációs Kutatóközpont,
171 oldal. Ára: 75,— Ft.)

A szótár készítői az utóbbi évek szakirodalmában mind gyakrabban felbukkanó számítógépes fogalmakat jól kezelhető szótárba gyűjtötték. Céljuk az volt, hogy a számítástechnika iránt érdeklődők, elsősorban a tanulóifjúság segítséget kapjon a folyóiratokban, gépismertetőkben, programnyelvekkel kapcsolatos leírásokban található fogalmak értelmezéséhez. A szerteágazó ismeretanyagból elsősorban a mikroszámítógépek és a személyi számítógépek tanulmányozása során előkerülő szakkifejezéseket emelték ki. A magyar—angol számítástechnikai kisszótár (tévesen az Angol—magyar számítástechnikai kisszótár címet viseli) a fogalmak rövid magyarázatát is megadja. Az angol—magyar szójegyzék elsősorban az angol szakirodalom olvasóinak nyújt segítséget a legfontosabb kifejezések értelmezésében.

Az olvasó a kötet végén külön összegyűjtve találja az adatátvitellel kapcsolatos angol—magyar szószedetet és a BASIC-nyelv utasításainak gyűjteményét.

Szlávi Péter—Zsakó László:
Módszeres programozás
(Budapest, 1986.
Műszaki Könyvkiadó,
116 oldal. Ára: 50,— Ft.)

A szerzők, miután konkrét példák alapján megfogalmazzák a programkészítéssel kapcsolatos problémákat, közérthető stílusban szólnak azok megoldásáról, a korszerű strukturált programozási módszerekről. Az elveket több példán keresztül mutatják be. Módszert adnak a programok helyességének bizonyítására, a hatékonyság növelésére. A könyvben ismertetett elveiknek megfelelően egy elvont, magyar alapszavakból álló magas szintű programozási nyelvet definiálnak, ebben írják programpéldáikat is, de a függelékben megadják a különböző BASIC-változatokra vonatkozó átkódolási szabályokat is.

Erdős Iván:
A COMMODORE 64 assembly nyelvű programozása
Budapest, 1986.
LSI ATSZ, 146 oldal.
Ára: 138,— Ft.)

A könyv közérthető módon ismerteti a C64 processzorának utasításkészletét és az assembly programozást segítő — hazánkban legelterjedtebb — programcsomagokat. Módszertani utasításokat is közöl, hogy az assembly nyelvet mikor és hogyan kell használni a C64 esetében.

MIKROSZÁMÍT
-FELHASZNÁLÓ

$ hely

1. ajánlatunk:

WINCHESTER DISC CONTROLLER
SCSI (SASI) interface
— max. 4 db tetszőleges típusú és kapacitású
— 5,25" Winchester disc drive vezérlése
— funkcionálisan kompatibilis a XEBEC S1410, ill. WD 1002-SHD vezérlőkkel

Ár: tartozékokkal 59 900 Ft

2. ajánlatunk:

IBM KOMPATÍBILIS FLOPPY DISC DRIVE CONTROLLER
Ár: tartozékokkal 18 900 Ft

Szállítás raktárról
12 hónap garancia

Centrumban

A Centrum Áruházak Vállalat egyszerre 30 áruházában kezdett forgalmazni mikroszámítógépet, mégpedig a Videoton által gyártott TV-Computert. Különösen jelentős az a tény, hogy az áruházak közül több mint 20 vidéken van, szétszórva az egész országban. Így végső soron a Centrum is aktívan részt vesz a számítástechnikai eszközök terjesztésén túl a számítástechnikai kultúra országos terjesztésében.

Spectrum – Commodore 64 kapcsolat

Kétféle ár—teljesítmény kapcsolatot épített ki a Háziszámítógép-építő Klub, a HCC a ZX-Spectrum és a Commodore 64 között. Az olcsóbb változat pár száz forintos áron 50 baud sebességű átvitelt biztosít a Spectrumból a C64-be. Képernyőtartalom és adatállomány egyaránt átküldhető közvetlen üzemmódban.

Mintegy tíz-hússzoros áron az átvitel sebessége is tíz-hússzorosára nő, és ez a megoldás már kétirányú kapcsolatot is biztosít.

-GYÁRTÓK,
GYELMÉBE
t!
ÁRENGEDMÉNY
nagyobb darabszámú
rendelés esetén
Felvilágosítás:
Gigor Györgyné
Telefon: 693-253
Levélcím:
1369 Budapest Pf. 317.
mikromatika

Lakossági számítógép-szerviz

1984 januárjában alakította meg az Agroinvesztria Innovációs Vállalat a Professional PC szervizt, elsősorban az SZKI által készített M08X és Proper típusú számítógépek javítására. Azóta profiljuk kibővült, és a háziszámítógépektől kezdve az IBM PC-ig mindennel foglalkoznak. Az igényeknek megfelelően, a közületi szolgáltatásokon kívül alig fél éve megalakult a lakossági szolgáltató részleg is, mely főképpen a Commodore 64 és a Sinclair Spectrum gépek javításával foglalkozik.

A Professional PC szerviz legfontosabb jellemzője a gyorsaság. Ennek érdekében a felhasználók közelébe kell települnie. Ezért hozta létre országos hálózatát: ma már Budapesten kívül hat másik városban is van 4-6 fős kirendeltsége. Ezek nem csupán rendelésfelvevő helyek, hanem maguk is javítanak. Nem utolsósorban e hálózatnak is köszönhető, hogy a szervizszerződést kötő intézményekhez Budapesten 24 órán, vidéken pedig 48 órán belüli kiszállást vállalnak.

Angliai telefonok

Erőteljesen növekszik a kereslet Angliában a kommunikációs eszközök, így a korszerű telefonok iránt. Jól jellemzi a helyzetet az ország telefonellátottságának alakulása: 1982 közepén 28,5 millió telefon volt az országban, azaz ezer lakosra 500 telefon jutott; az előrejelzések szerint 1988-ban már 40 millió készülék lesz.

A telefon ugyanakkor egyre intelligensebbé válik, egyre többet tud. Ez elsősorban az üzleti célokat szolgáló 10 millió készülékre vonatkozik, melyek közül az újak már számítógéphez csatlakozva címtárakhoz, határidőnaplókhoz is kapcsolatot biztosítanak. További jelentős fejlődés várható, ha a nyilvános hálózatban a jelátvitel teljes egészében digitálissá válik.

Keresztnevek

Az ifjú szülők jól tudják, milyen nehéz feladat az újszülött gyermek számára megfelelő nevet választani. Általában rengeteg fejtörés, tanácskozás, a nagyszülők, a rokonság, az ismerősök, a barátok széles körű bevonása szokta megelőzni a névadást.

Ezeken a gondokon kíván most segíteni Franciaországban egy újfajta szolgáltatás. A dél-franciaországi Annecyben megnyílt egy találékony fiatalember, a 28 éves Claude Bouvier irodája, amely számítógép bevonásával siet a szülők segítségére. Begyűjti az újszülöttre vonatkozó adatokat, a szülők kívánságait, a rokonság, a barátok véleményét, az általuk preferált neveket és mindenféle információt, többek között a gyermek nemét, a szülők családnevét, azt a kívánságukat, hogy milyen hosszú legyen a keresztnév, összhangban legyen-e a családnévvel, modern legyen-e vagy régi vágású stb. Mindezeknek az adatoknak a bevitele után a számítógép tíz keresztnevet terjeszt elő, ezek közül választhatják ki a szülők a nekik legmegfelelőbbet.

Árakkal

1985 őszén fejlesztette ki a Besztercebányai Számítógépgyár Csehszlovákia első, az IBM PC-vel kompatibilis mikroszámítógépét, a PP—06A-t. Egy évvel később, 1986 őszén ugyanitt készült el Csehszlovákia első, az IBM PC XT-vel kompatibilis típusa, a PP—06B. Újdonságot jelent, hogy az ezeket a típusokat hirdető prospektuson már ár is található: az A változat 100 ezer, a B változat 150 ezer koronába kerül (220 ezer és 330 ezer forint). Ez még sokáig eszmei ár marad, mert az idén csupán 50, 1988-ban pedig az A változatból 100, a B változatból pedig 300 darab gyártását tervezik.

A gépekhez kétféle operációs rendszert adnak: a PP—DOS az MS—DOS, a MIKROS—86 pedig a CP/M—86 megfelelője.

A PP—06 típusú mikroszámítógép

Kardiológia

A szovjetunióbeli Kisinyovban a szívbetegséggel való küzdelem hatékonyságának javítására központi kardiológiai adatbázist hoztak létre, mely a betegek személyi adatain kívül kórtörténetüket és néhány legutolsó EKG-jüket is tartalmazza. Az utóbbiaknak a számítógépbe való továbbítására a városi telefonhálózatot használják. Az orvosokat ellátták egy hordozható, az EKG mérésére és továbbítására szolgáló, Volna nevű készülékkel.

Az EKG-felvétel a következőképpen történik. Az orvos akár a beteg lakásán is elkészíti az EKG-t, majd közvetlenül a beteg lakásából felhívja a kardiológiai központ számát. Bediktálja a diszpécsernek a beteg azonosító adatait, majd a telefonra egy tapadókorongot helyezve, a Volna típusú készülék továbbítja az EKG-mérés eredményét. A vételt a hierarchikus rendszer alsó szintjén lévő Elektronika—60 típusú mikroszámítógép végzi.

Ez végrehajtja a zavarszűrést, a zavarok kiegyenlítését, továbbá kijelöli az EKG-görbén a jellemző szakaszokat, majd feladja az így előkészített görbét egy SZM—4 típusú minigépre. Ez behelyezi az új EKG-t a beteg már meglévő adatai közé, illetve új beteg esetén felviszi a személyi adatokat és elvégzi a „betegfelvételt". Ezután a friss EKG, továbbá a rendelkezésre álló kórtörténet alapján egy relációs szervezésű tudásbázis felhasználásával az orvosszakértői rendszer diagnózist készít. Mindez néhány másodpercet igényel csupán. Ezért a diszpécser az EKG-vizsgálat után az előtte lévő képernyőről azonnal beolvassa a telefonba az orvos számára a diagnózist.

Az 1985 óta sikeresen üzemelő rendszerbe évente 6-7 ezer EKG-mérés érkezik be. A kezelési hatékonyság javulásának eredményeiről az eltelt időszak rövidsége miatt még korai beszélni, de mind a betegek, mind az orvosok ma már nélkülözhetetlennek tartják az új rendszert.

Adatmentők

Kávé vagy üdítő ömlik a hajlékonylemezre — tipikus esemény az irodai alkalmazásoknál. Ám a lemezen lévő információ ilyenkor elvész, ami sokszor pótolhatatlan kárt okoz. Ennek elhárítására készült fel a hajlékonylemezeket is gyártó svéd Kopparberg cég. Az ősszel Budapesten tartott bemutatójukon érdeklődők nagy tömege követte végig a mosásból, vegyszeres kezelésből és a borítékjából kivett hajlékonylemez felületi csiszolásából álló mentési technológiai folyamatot. Az eredmény: az IBM PC ismét el tudta olvasni a lemezen lévő információt, így az átmenthetővé vált egy másik, új lemezre.

A bemutató során elmondták, hogy felkészültek a „gyengén" felírt lemezek olvasására is. Ez a probléma általában a lemezek többéves tárolása után jelentkezik: váratlanul kiderül, hogy a mikroszámítógép már nem képes elolvasni a felírt információt. Ekkor a laboratóriumukban egy rendkívül érzékeny géppel olvassák végig a lemezt, amely az igen „halk" jelet is képes olvasni. Ez azonban a „zajok" felerősödéséhez is vezet. A zajt a technológia következő lépcsőfokán kiszűrik.

A Kopparberg cég adatmentési szolgáltatása saját gyártmányú lemezei esetében Svédországban ingyenes. Már folynak a tárgyalások egy magyar-svéd vegyesvállalat létrehozására, mely előreláthatólag az év közepén hazánkban is megnyitja adatmentő szolgálatát.

Spectrum + CP/M

A lengyel Nevelés- és Tudományügyi Minisztérium tavaly iskolaszámítógép-pályázatot hirdetett. A megadott követelményrendszer első pontja volt, hogy a pályázó gépnek kompatíbilisnek kell lennie a Lengyelországban jelenleg legjobban elterjedt ZX-Spectrummal. A többi követelmény között találjuk a szocialista gyártmányú alkatrészbázis igényét, a lokális hálózatba kötés lehetőségét a szaktantermek kialakíthatóságához, a jó kiépíthetőséget (hajlékonylemezes tároló, lengyel gyártmányú nyomtató csatlakoztathatósága), a maximum 100 ezer zlotys árat s az akár évi százezer darab gyártására is képes kapacitást.

A pályázatot az Elwro 800 Junior nyerte, ami egyben az 1985-ben bemutatott Elwro 700 háziszámítógép gyártáselőkészítésének leállítását is jelentette. A Junior nem csupán a Spectrummal kompatíbilis, hanem a CP/M operációs rendszer is fut rajta. Mindkét üzemmódjában képes kezelni a hozzá csatlakoztatott hajlékonylemezes tárolót és nyomtatót. Az új iskolaszámítógépet 1986 nyarán a poznani vásáron mutatták be elsőként a nagyközönségnek.

Az Egyesült Államokban minden évben rendeznek néhány nagyobb számítógép-tudományi konferenciát, kiállítással egybekötve, az ország különböző részein. Ezek egyikét, a Nemzeti Számítógép Konferenciát (National Computer Conference — NCC '86) 1986-ban a Nevada állambeli Las Vegasban tartották, június 16-tól 19-ig.

Az NCC-t ezúttal a legrangosabb szakmai társaságok védnöksége alatt tartották meg. A konferencia jelszava: „A számítástechnika dinamikus kiterjesztése a vezetők, a szakemberek és a felhasználók körében" volt. A szervezők azt mondták, hogy ez lesz a számítástechnika „rendezvényeinek rendezvénye" (Show of Shows).

A rendezőség igyekezett is a résztvevőknek mindent megadni; például a szállodák és a néhány kilométerre levő kiállítási csarnok között ingyenes autóbuszok álltak rendelkezésre, volt üzenetközvetítés számítógépes hirdetőtáblán, ezenkívül a Computer World és egy másik, számítógépeket forgalmazó szaklap ingyenes napi különkiadásban jelent meg. Mindezek ellenére ez a rendezvény nem volt annyira sikeres, mint az előzőek.

Az 1973-as, első NCC, melyet New Yorkban tartottak, a „mainframe"-ekkel foglalkozott, és közönsége inkább a tudományos kutatók közül került ki. Mostanára azonban megváltozott az eredeti cél, és az utóbbi konferenciának a nagyközönséget igyekeztek megnyerni. Az üzletemberektől a felhasználókig terjedő spektrumban a téma viszont túl általánossá vált. A sikertelenség fő oka mégis az volt, hogy az elmúlt hónapokban már több hasonló, de jobban specializált kiállítást rendeztek. Rövid idő alatt nem lehet sok újat látni, és mind a kiállítók, mind a látogatók kezdtek belefáradni a sok rendezvénybe.

A kiállítók száma ezúttal mindössze 400 volt a néhány évvel korábbi közel ezerrel szemben, a 25 ezer látogató pedig csupán az egynegyede a négy évvel ezelőttinek. Más kérdés, hogy akkoriban a rendezvény túlméretezettségét panaszolták.

Az egyetlen szenzáció az volt, hogy az IBM ezt a kiállítást választotta új gyártmányainak bejelentésére. Néhány nagyobb amerikai hardver cég (Digital Equipment, Data General, Apple, Compaq) és mikroszoftver-vállalkozás (Ashton-Tate, Lotus, Microsoft) távol maradt az NCC-től. Ezzel szemben, főleg a japán cégeknek köszönhetően, a nemzetközi kiállítók és a nemzetközi látogatók aránya nőtt az előző évekhez képest.

Egy egész napot fordítottak a konferencián a szakemberek és az egyetemi hallgatók találkozására. A tapasztalatátadási lehetőség abban segítette a fiatalokat, hogy végiggondolják: hogyan kell megtervezni az előmenetelt az informatikai szakmában.

Hangsúlyozták, hogy csak azok a számítástechnikai vállalkozások lesznek sikeresek, melyek a megoldásra összpontosítanak, és csak azok a felhasználók, akik az információt a stratégiájukban képesek alkalmazni.

Az NCC-n első ízben osztottak ki elismerő okleveleket az elmúlt évben kiváló eredményt elért három szakembernek és három gyártmánynak. Ez utóbbi kategóriában a Plus Development Corporationt jutalmazták a 10 megabájtos „Hardcard"-ért, a Telos Software Products céget a „Business Filevision" adatrendszerért és a Teradata Corporationt a DBC/1012 „Database Computer System"-ért.

Az NCC '86 fénypontjaként említett IBM-bejelentésben a több mint 125 új vagy továbbfejlesztett gyártmány közül a következők voltak a legjelentősebbek:

System/36. Új modell gyorsabb feldolgozóegységgel, nagyobb központi tárral és kiépíthetőséggel. Bővült az irodai alkalmazások lehetősége, és továbbfejlesztették az operációs rendszert is.

System/38. Újabb modell, nagyobb teljesítmény.

Összekapcsolt System/36 — System/38. A berendezések egymás erőforrásait képesek használni.

Új megjelenítő rendszerek. Displayk (IBM 3171, 3190, 3191 stb.), grafikus beviteli egységek (IBM 3117 és 3118).

„InfoAblak" (InfoWindow). IBM PC érintő képernyővel, amely videolemezzel összekapcsolva sokféle információ párbeszédes megjelenítésére alkalmas.

Displaywrite 4. A szövegfeldolgozó szoftver feljavított változata, amely a PC és System/36 közötti „seamless" összakapcsolásra alkalmas.

Az **IBM PC RT** modellt is bemutatták, amelynek megnövelték a teljesítményét, megváltoztatva a hardvert és a szoftvert. A rendszer gyűrűs hálózatba kapcsolható.

A másik jelentős bejelentést az American Telephone and Telegraph (AT&T) tette; piacra hozta ugyanis a régóta várt Unix System V operációs rendszer 3.0 változatát. Ezzel különböző rendszerű számítógépeket lehet összekötni az adathálózaton. Megemlíthetjük még a Ricoh System cég két új, percenként 40 oldal teljesítményű lézernyomtatóját, a Leading Edge új modemjét és PC/AT rendszerű mikroszámítógépét, a NEC Information System miniszámítógépeit, a Zenith Data Systems és a Toshiba cég ölbe vehető, hordozható számítógépeit, valamint a Fujitsu cég dokumentumolvasó egységét.

A tudományos kutatási és fej-

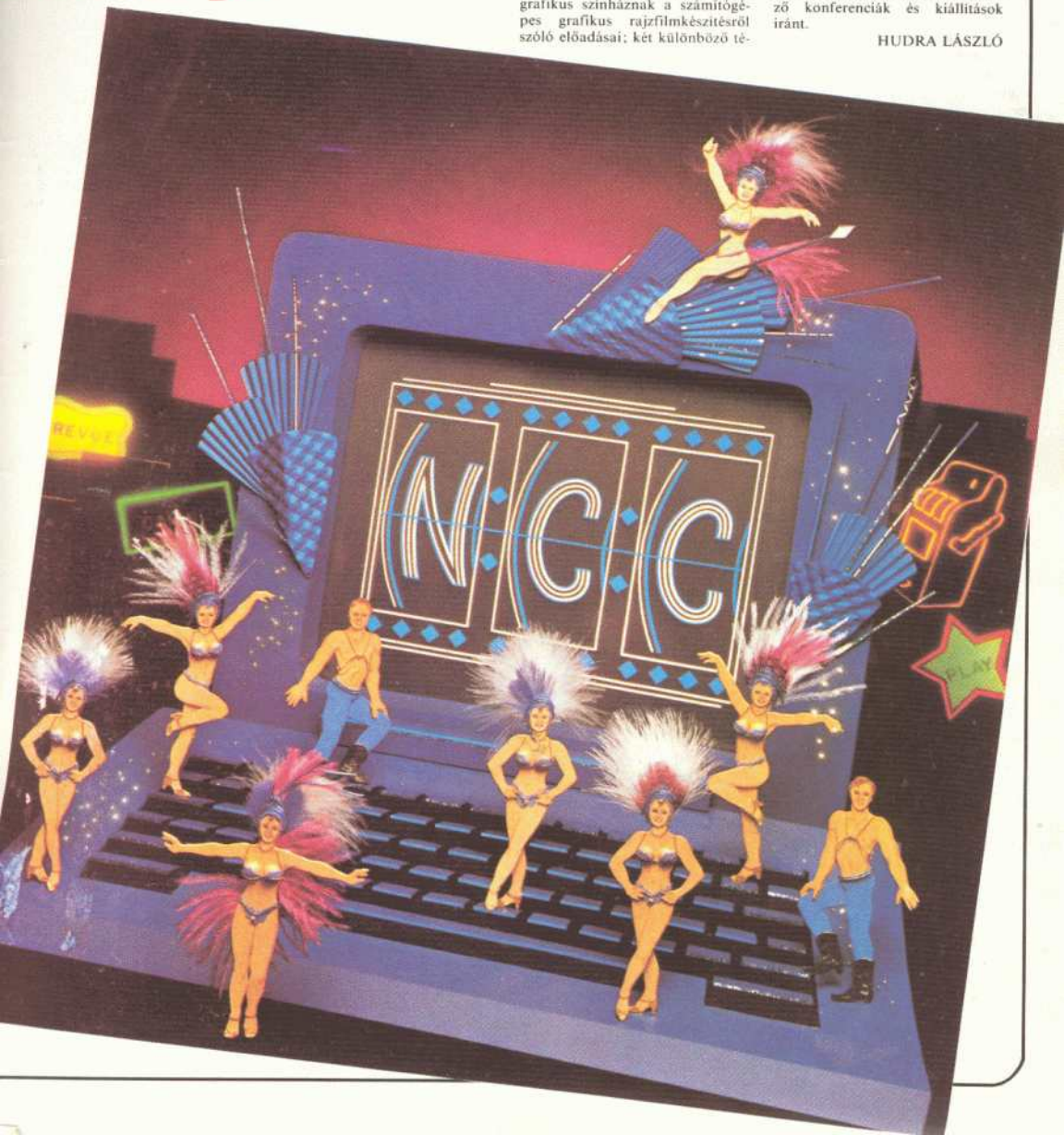
NCC'86
A világ legnagyobb show-ja

lesztési témák kiállítását külön tartották. Ezen több amerikai és egy kanadai egyetem, valamint kutatóintézetek vettek részt. Itt mindennap tájékoztatták az érdeklődőket a különböző számítástudományi és technológiai kutatási eredményekről. Az utolsó napon rendezték meg a kutatás és fejlesztés napját, továbbá a kisebb üzleti vállalkozások napját. Ez utóbbin a számítógépek alkalmazásának előnyeit és problémáit vitatták meg.

Igen érdekesek voltak az ún. grafikus színháznak a számítógépes grafikus rajzfilmkészítésről szóló előadásai; két különböző témát mutattak be, naponta kétszer. Látni lehetett egy háromdimenziós képet, amikor is a vetítővászonról kinyúló kígyószerű tűnemény szinte a nézők feje felett lebegett.

Erről a kevésbé sikeres NCC-ről is a legtöbb látogató valószínűleg megelégedéssel távozott. A társaságok máris szervezik és hirdetik a további speciális kiállításokat és a Chicagóban rendezendő NCC '87-et. Az NCC '86 tapasztalatainak felhasználásával bizonyára fokozni tudják az érdeklődést a következő konferenciák és kiállítások iránt.

HUDRA LÁSZLÓ



Tudomány
és fantázia?
Mandelbrot
halmazok
PTA 4000-rel